THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR	Katy Campbell-Bonar
TITLE OF THESIS	Understanding Success in Computer Programming

DEGREE FOR WHICH THESIS WAS PRESENTED M. Ed.

YEAR THIS DEGREE GRANTED 1984

[illegible]

THE UNIVERSITY OF ALBERTA

UNDERSTANDING SUCCESS IN COMPUTER PROGRAMMING

by

© Katy Campbell-Bonar

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF EDUCATION

DEPARTMENT OF SECONDARY EDUCATION

EDMONTON, ALBERTA

SPRING, 1984

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **UNDERSTANDING SUCCESS IN COMPUTER PROGRAMMING** submitted by Katy Campbell-Bonar in partial fulfilment of the requirements for the degree of Master of Education.

ABSTRACT

A study was undertaken to identify factors influencing success in programming in BASIC at the junior high school level.

Seven classes of Grade 7 students (N=158) were identified at Kenilworth Junior High School in Edmonton, Alberta. None of the students had had formal instruction in programming in BASIC.

Based on a study by Naiman and others for the Ontario Institute of Studies in Education (1978) which described the Good Language Learner, several measures of cognitive style and personality traits were identified for inclusion in this study. Initially, the <u>Group Embedded Figures Test (GEFT)</u> (Witkin, et. al, 1971) and <u>Budner's Scale of Tolerance/ Intolerance of Ambiguity</u> (Budner, S., 1962) were chosen to be administered to all the participants prior to programming instruction.

Because of vocabulary and comprehension difficulties, the Budner scale was consequently eliminated, based on a trial administration of the measure to a random selection of students.

One week prior to the beginning of the instruction period (approximately seven weeks) all the participants completed the GEFT. Subsequently, twelve additional variables were identified that would be likely to contribute to success in learning a programming language. They were: age; sex; scores on measures of verbal, non-verbal and

quantitative IQ; scores on reading-decoding and reading-comprehension of a system-wide comprehensive reading examination; teacher-assigned term marks in language arts, mathematics, science, social studies and French; and enrollment in a French course.

After approximately 360 minutes of instruction in BASIC all participants were asked to complete two programming assignments: one involving the writing of a program that was to meet certain criteria, and the second involving the interpretation of a program listing.

An analysis of the data involving a multiple correlation and a multiple regression analysis revealed that eleven variables qualified for consideration for the first test of programming ability (IQV, IQQ, IQN, RD, RC, LA, M/S, SS, F, GEFT, P2) and six qualified for consideration for the second test of programming ability (IQN, IQQ, RD, LA, GEFT, P1).

ACKNOWLEDGEMENTS

Without the unfailing support of Dr. Douglas V. Parker, this study would not have been undertaken. Not only did he set high standards, but he was constantly encouraging when the completion of this thesis seemed an unattainable goal.

Likewise, I would like to thank Dr. James LaFollette and Dr. George Cathcart for their interest in this study.

Carl Nishimura provided the initial access to the Grade 7 students at Kenilworth Junior High School and "smoothed my way" with the teachers and students there. Without Carl's interest, advice and flexibility this study would not have been successfully completed.

I would like to say a special "thank-you" to my family: to my parents who always expected high levels of achievement from their daughters and were nevertheless accepting of individual differences; to my sisters who were inspiring; and to my husband who, quite apart from his extraordinary patience with my studies, has remained proud of my work.

Finally, I am extremely grateful to Pat Pasos, my typist, for her beautiful, exacting work which never varied through many revisions.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER I

Introduction

In a report made available in January 1982, the Computer Literacy Steering Committee (A committee of Alberta Education, the Province of Alberta) discussed what it took to be the three dimensions of computer literacy: Awareness, working knowledge of function, and critical understanding.

Recognizing the "pervasive and intrinsic nature of computer activity within our society" (Leverett, 1978) and stressing the need for all students to have equal opportunity of gaining understanding in this area, the Committee further notes that the optimum time to teach computer literacy is before students make critical option choices in high school.

Recommending that the elementary school unit deal with overall awareness of computers in society, the Committee suggested that the junior high school level emphasize a "working knowledge" of computers, the second dimension of computer literacy. These recommendations were implemented in a pilot program in September, 1983, and a computer curriculum with the orientation suggested by the report may ultimately involve _all_ students at the junior high school level.

The recommendations of the Steering Committee and the imminence of all junior high school students becoming

involved in a computer literacy program make it desirable to find or devise a study, the results of which can be used as a guideline in helping educators to predict student success in computer programming. Such a study would be useful in determining the most effective teaching techniques for both types of students identified: those likely to be successful programmers and those likely to experience difficulty in conceptualizing programming problems. The former group might, for example, proceed better if assigned all their problems at once and encouraged to work individually, while the "problem" programmers might better benefit from a formal structured teaching approach (Cheney, 1980)[1].

A search of the literature reveals that there are two instruments designed specifically to predict success in computer programming: The Programmer Aptitude Test (PAT), developed for the International Business Machine Corporation; and the Robot Test, developed by the Bureau of Census, United States Department of Congress. Both tests, however, present serious shortcomings when considered for use at the junior high school level in the Alberta system. Both tests are designed for use with populations that are college age or older, and neither test is well standardized, the latter being in its eighth revision (Howell, M.A., Vincent, J.W.,

1 For a discussion of principles of designing instruction for specific cognitive styles, see Ausburn, L.J. & Ausburn, F.B., Cognitive Styles: Some Information and Implications for Instructional Design, ECTJ, Vol. 26, No. 4, pp 337-354, Winter, 1978.

and Gary, R.A., 1967). Further, both tests have been developed in the American context ultimately for business applications and show positive correlations with age, education, Civil Service Grades and, in the case of the PAT, supervisory reports not related to programming. In fact, Howell (1967) suggests that a vocational aptitude test might be better suited for predicting success in programming. The problem with using or adapting either of these tests to serve as the kind of instrument I have described suggest that other tests are needed for this purpose.

Computer programming languages display many of the features that characterize natural languages, for instance vocabulary and syntax (Sammet, 1969). In addition, there is already evidence that successful language learners and computer programmers exhibit similar cognitive styles (Cheney, 1980; Naiman, N., Stolick, M., Stern, H.H. and Todesco, A., 1978). Given these similarities, it is possible that successful second language learners may be successful programmers and can be identified as such.

This study, therefore, primarily will explore the relationships among these factors: cognitive styles associated with success in language learning and in learning programming languages, general academic performance, and other factors that may be related to either of the above (for example, age and sex).

Statement of the Problem

The general formulation of the question of this study is: What factors related to learner characteristics will allow us to predict success in computer programming in BASIC at the junior high school level in Alberta?

The study specifically will address this question:

Which of the following or combination of the following will allow us to predict success in computer programming at the junior high school level in Alberta?

1) Score on the _Group Embedded Figures Test_.
2) Position of the _Budner's Scale of Tolerance/Intolerance of Ambiguity_.
3) Score on the _Reading Survey Test (EPSB)_.
4) I.Q. scores (verbal, non-verbal, quantitative) as measured by _Canadian Cognitive Abilities_.
5) French score.[2]
6) Term marks in the major academic subjects: Language, Arts, Social Studies, Mathematics, Science.
7) Sex.
8) Age.

Categories (1) through (6) deal with standard measures of language learning aptitude and associated cognitive styles. Categories (7) and (8) act as a check on the possible generality of the results.

---

2 This will be a percentage grade assigned by the French teacher for the previous term's work in French language arts (i.e. the study of the nature of the French language, including syntax and vocabulary).

Major Assumptions of the Study

1. A teacher-assigned grade in an academic core subject is an accurate measure of performance in that subject.
2. A teacher-designed measure of success in programming is an adequate measure of success in learning certain aspects of BASIC.
3. A short course in programming in BASIC as devised by the researcher is comprehensive enough to provide a basis for measuring success in programming.

Definition of Terms

The following terms will be used in this study as defined below:

Computer Literacy: At the junior high school level, a functional or working knowledge of computers and their problem-solving capabilities (Computer Literacy Committee, 1982) within the context of a sociological understanding of computers in society.

Programming Language: "A general term for a defined set of symbols plus the rules or conventions governing the manner and sequence in which the symbols may be combined into meaningful communication" (Sammet, 1969, p. 8). In this study, the programming language is BASIC.

Learner Characteristics: Characteristics likely to in-

fluence the choice and use of learning strategies and techniques (Naiman et al, 1978). In this study, learner characteristics include cognitive style, intelligence and academic performance.

Cognitive Style: Problem solving methodology employed by an individual in a decision situation (Cheney, 1980). In this study, cognitive style will be limited to field dependence/independence as measured by the Group Embedded Figures Test, and tolerance/intolerance as measured by the Budner's Scale of Tolerance/Intolerance of Ambiguity.

Aptitude: Carroll (1963) defines aptitude as an inverse function of the time required by an individual school, university and adult level to achieve a certain level of mastery in a second language: in this study, competence in the programming concepts taught by the researcher (Carroll, in Krashen, S.D., 1981).

Procedural Outline of the Study

This study involved Grade 7 students attending a junior high school in Edmonton, Alberta. The study involved the following procedures:

1) All students were administered:

   a) The Group Embedded Figures Test (1971)

   b) Budner's Scale of Tolerance/Intolerance of Ambiguity (Budner, S., 1962)

2) Scores from the following measures were obtained:

   a) _Canadian Cognitive Abilities_ (1974)

   b) _Reading Survey Test_ (EPSB)

3) From cumulative records, the following scores were obtained:

   a) Grade 7 French scores for the previous reporting period.

   b) Academic core subjects (math, science, social studies, language arts).

4) Sex and age of all participants was noted.

5) All participants (N=158) participated in a short course in microcomputing (approximately 360 minutes) taught by the researcher and observed by other teachers.

6) The following analyses were performed on the resulting data:

   a) Calculation of the mean, media and quartile for three categories of success in programming: _Excellent_ (E); _Satisfactory_ (S); and _Needs Improvement_ (N).

   b) Multiple correlations on all pairs of variables with a pair-wise exclusion of missing data.

   c) Multiple regression analysis for a step-wise prediction for two measures of programming ability.

This final step in the procedure should allow one to predict performance in computer programming from several optimally combined independent variables (Hopkins, K.D. and Glass, G.V., 1978).

Delimitations of the Study

1) This study is restricted to students in Grade 7 at Kenilworth Junior High School in Edmonton.
2) This study is restricted to students who have not been involved in a microcomputer option.
3) Although many factors and learner characteristics are thought to be important to language learning, this study will be limited to those factors and learner characteristics previously outlined.

Limitations of the Study

1) Since the selection of subjects for this study was non-random, the results may not be generalized to other groups.
2) Since success in computer programming will be judged by the researcher and the teacher of the option, the standards set for assignment to the categories of Excellent (E), Satisfactory (S), and Needs Improvement (N) will be subjective and not based on a standardized measure.

Significance of the Study

A study by Dolotia, Bernstein, Dickson, France, Rosenblatt, Smith and Steel (in Lemos, R.S., 1980) predicts that by 1985 70% of the labour force will rely on data processing in the course of their daily job-related activities. They will, therefore, require an understanding of data processing technology and its related applications. The study further indicates that the majority of these workers will not be professionals in this field.

With society's increasing reliance on computer activities, it seems evident that our attention as educators should be directed to the teaching and learning processes involved in computer programming. Our awareness of the factors that influence success in computer programming will help us to identify learning obstacles, to compare alternative methods of instruction, and to provide constructive feedback to our students (Lemos, R.S., 1980).

These concerns should be especially acute for Alberta educators as courses in computer literacy may become compulsory at the junior high school level in Alberta after being piloted in September, 1983.

CHAPTER II

Review of the Related Literature

There are a number of studies that suggest the desirability of focusing on aptitude for language acquisition in attempting to isolate predictors for success in the acquisition of computer programming skills.

Language Aptitude

A. Carroll and Language Aptitude

John B. Carroll (1963) developed a conceptual model of second language learning in which he identified five factors as governing second language success: Aptitude, general intelligence, perserverance, quality of instruction, and opportunity (p. 1060). In keeping with the focus of this study, I shall deal only with the first of these factors.

Carroll (1963) defines aptitude as an "inverse function of the amount of time required to attain a criterion mastery in the task to be learned" (p. 1061). As such, it is a function of a number of independent basic traits or characteristics of the learner.

In the 1950's Carroll had identified three components of language aptitude: Phonetic coding ability, grammatical sensitivity, and inductive learning ability. These latter two components seem to have special significance for learn-

ing a programming language, since no oral component is involved.

Grammatical sensitivity is the ability to understand grammatical functions of different kinds of language elements (Carroll, 1977). Noting that this component related to grades in general and to academic achievement outside the language classroom, Gardner and Lambert (in Krashen, S.D., 1981) suggest that language aptitude is not related solely to achievement in the second language classroom.

Inductive ability, which is the ability to "examine language material ... and from this to notice and identify patterns, correspondences and relationships" (Carroll, in Krashen, S.D., 1981), typically involves presenting materials in an artificial language so as to provide the opportunity of inducing grammatical and semantic rules governing the language. An inductive approach to the discovery of a set of explicit, abstract rules seems an obvious way to test programming language aptitude as well; this skill is indeed tested by instruments such as the _Robot Test_ (Howell, M.A. et al., 1967).

In reviewing his own and other's research into language aptitude, Carroll came to three general conclusions worth quoting here (Carroll, 1963, pp. 1088 ff):

> It is possible to predict success in intensive language courses with high validity by means of certain tests.

> FL aptitude is not specific to particular groups of languages, the same battery of tests predicts success in languages as diverse as German and Chinese with approximately the same degree of validity.
>
> Some evidence indicates that a battery of language aptitude tests can provide information useful in forecasting and diagnosing particular types of learning difficulties.

In relating Carroll's conclusions to this study, it seems reasonable to suggest that those students identified as being successful in French would be successful at learning a programming language; and that a measurement developed to predict aptitude in one would be suitable for predicting aptitude in the other.

B. Other Studies of Language Aptitude

Pimsleur, Stockwell and Comrey (1961, p. 15ff) described aspects of two of their studies which attempted to identify variables hypothesized to be related to success in second language learning. The first study, consisting of 21 test variables, omitted several factors recognized to be of potential importance in language aptitude including auditory discrimination and language interests. These factors appeared in the second study. The first study, described here, has particular relevance for aptitude testing in programming languages as it includes variables related to factors other than proficiency in auditory and verbal comprehension.

The 23 variables providing data for the study were as follows:

1. MLAT[3] - Spelling Clues
2. MLAT - Number Learning
3. MLAT - Words in Sentences
4. MLAT - Paired Associates
5. Letter Series - Guilford
6. Reading Aloud I - Speed
7. Reading Aloud I - Accuracy
8. Reading Aloud II - Speed
9. Reading Aloud II - Accuracy
10. Paraphrase
11. Rhymes
12. Synonyms
13. Phonetic Perception
14. Linguistic Analysis I
15. Linguistic Analysis II
16. Verbal Comprehension
17. Age
18. Sex
19. Bilingualism
20. High School Language Grades
21. High School Math/Science Grades
22. French Final Grades
23. French Speaking Proficiency

An analysis of multiple correlations yielded a basic battery which minimized the number of tests while maximizing

3 Modern Language Aptitude Test

the multiple correlation coefficient, r. The results are shown in Table I.

For predicting French grades, the seven-test battery includes Number Learning (or Spelling Clues), Words in Sentences, Letter Series, Reading Aloud I, Paraphrase, Linguistic Analysis II, Age, and High School Math/Science Grades. Aspects of this basic battery will be used in this study to predict success in computer programming.

Although a search of the literature confirms that prior aptitude testing predicts success in foreign language learning, one conclusion drawn from several early studies in aptitude testing has not altered to date: Nothing can predict success or failure as reliably as an actual tryout in the foreign language (Harding, 1958). McEwan (1974) supported this finding by indicating the best single predictor of performance in French appears to be actual performance. Ideally, then, one might seek a situation in teaching programming languages where all students would be admitted to the option prior to (or in lieu of) aptitude testing. After a trial period in which they were carefully observed, the students could be routed to the teachers whose approaches most closely matched their various learning styles.

Table I. Multiple Correlation Analysis

Variable	French II Grades		French Speaking Grades	
	A (21 tests)	B (7 tests)	C (21 tests)	D (5 tests)
MLAT - Spelling Clue	.033		.058	.058
MLAT - Number Learning	.025	.036	.002	
MLAT - Words in Sentences	.047	.063	.007	
MLAT - Paired Associates	.002		.003	
Letter Series - Guilford	.008		.019	.024
Reading Aloud I - Speed	.021	.029	.023	
Reading Aloud I - Accuracy	.000		.003	
Reading Aloud II - Speed	.002		.010	.027
Reading Aloud II - Accuracy	.014		.001	
Paraphrase	.009	.012	.001	
Rhymes	.001		.000	
Synonyms	.000		.000	
Phonetic Perception	.005		.008	
Linguistic Analysis I	.000		.000	
Linguistic Analysis II	.010	.012	.004	
Verbal Comprehension	.002		.036	.044
Age	.012	.012	.002	
Sex	.001		.001	
Bilingualism	.022	.024	.028	.023
High School Language Grades	.004		.000	
High School Math/Science Grades	.008		.001	
	R=.478	R=.433	R=.454	R=.418

A & C include all tests
B & D include most economical battery

The Good Language Learner

The student who is successful in learning a second language, dubbed the "Good Language Learner" (GLL), by Stern (1975), Krashen (1977) and Naiman et al. (1978), appears to utilize well-documented strategies of language acquisition attributable to aspects of each student's personality and cognitive learning style.

Stern (1975) identifies strategies chosen by the GLL, after first providing an analysis of language acquisition. The analysis is meant to provide an answer to this question: What is involved in learning a second language?

Learning a language involves establishing a new reference system and formulating hypotheses (creativity) that allow ever closer approximations to the "new" reference system. It initially requires mechanical skill or skill in manipulating patterns. The language learner advances through extrapolation and application of these patterns to automatic production (Stern, p. 309). Stern indicates that this process gives rise to three major problems. Discrepancy between the learner's first language and the language being learned will be a dilemma which the poorer learner will fail to resolve. Secondly, the GLL must learn to attend to form and meaning simultaneously. This is labelled the code-communication dilemma. Thirdly, the language learner must choose among learning strategies in order to approach the task in one of three ways: Intellec-

tually, conceptually, or systematically. Using the foregoing analysis of language learning and its associated difficulties, Stern has identified ten learning strategies exhibited by good language learners. The good language learner has:

1) A positive learning strategy.
2) An active approach.
3) Tolerance and empathy towards native speakers.
4) Technical "know-how".
5) Strategies to "order" the task.
6) A constant curiosity about the "meaning" of items in the language.
7) A willingness to practice.
8) A willingness to "use" the language in real situations.
9) Critical sensitivity to language use.
10) The ability to develop the language as a separate reference system.

Carroll (1977) would agree with these strategies and add that a tolerance for ambiguity and seeming irrationality also characterizes the good language learner (Naiman, N. et. al, 1978).

Testing For the Good Language Learner

In 1978, a study was undertaken for the Ontario Institute of Studies in Education (OISE), in which the Good Language Learner was profiled. Naiman, et al. (1978) posed

the question "Do good learners tackle the language learning task differently from poor learners, and do learners have certain characteristics which predispose them to good or poor learning?" (p. 2). Suggesting that characteristics of the good language learner would include cognitive factors such as intelligence and language aptitude, cognitive style and personality factors, and attitude and motivation, the study described by Naiman investigated the relationship among these factors by the administration of various instruments.

A population of students at a Toronto high school, identified as successful language learners, were administered a battery of tests including the _Hidden Figures Test_, the _Strop Phenomenon: Speed of Color Discrimination Test_, _Pettigrew's Category-Width Scale_, _Budner's Intolerance of Ambiguity Scale_, and _Mehrabin's Sensitivity to Rejection Scale_. Field-independence, as measured by the _Hidden Figures Test_, and a high tolerance for ambiguity, as measured by Budner's scale, were the two characteristics that correlated most highly to scores on the International Education Association Tests of French Achievement.

Fromme (1980) agrees with Naiman, Frolich, Stern and Todesco (1978) that field independence and field dependence are important styles in language learning. Brown, however, with Krashen (1977), differentiates between child and adult language acquisition. It seems likely that adults would use more conscious attention to form than children, who tend to

"acquire" language subconsciously. Brown makes a further distinction between tutored language learning situations and natural language learning situations. It is important, in any case, to recognize field-independence/dependence as styles on a continuum that may be variable for any one person.[4]

The degree to which an individual is willing to tolerate ideas or systems contrary to his/her own was measured by _Budner's Intolerance of Ambiguity Scale_ in the OISE study. A significant finding was that tolerance of ambiguity is an important factor in second language learning.

Programming Languages

A. _Programming Languages and Natural Languages_

Sammett (1969) identifies the following functional characteristics of a programming language. Natural languages can also be analyzed in terms of these characteristics.

1) _Generality/Simplicity_ - generality refers to the ability of the language to apply directly to a wide class of problems and simplicity refers to the ease of learning, use and implementation.

4 An individual who displays field independence as a cognitive style tends to think analytically rather than globally.

2) Core Language Concept - generality and simplicity exist together and the user, building on a simple framework is able to handle a large number of problems.

3) Succinctness/Naturalness - succinct notation may be most natural [e.g. (p 31) ADD A to B & MULTIPLY that result by C to PRODUCE D (natural) vs D = C x (A + B) (succinct)].

4) Consistency - this involves the constant application of the same rules in the same way throughout the language.

5) Efficiency

6) Ease of Reading/Writing

7) Dialects - minor variations on a particular language for purposes of modification, improved efficiency, etc.

In addition, Sammet recognizes that programming languages possess syntax in the sense of legal character sequences, semantics in that legal strings may have a great many uses, and the elements of operators (connectives), delimitors (define beginning and end of sequences), punctuation, and noise words (which may be inserted or omitted at the users' option). All such features are also found in natural languages.

[illegible]

B. Predicting Success in Programming

Lemos (1980) makes reference to these shared characteristics of programming languages when discussing the difficulty of measuring programming language proficiency. He focuses on three dimensions of programming ability: Knowledge of language rules (grammar), ability to read programs (reading), and ability to write logically and gramatically correct programs. Lemos points out that measuring proficiency in these areas may be problematic owing to the uncertain relationship among these dimensions.

When teaching a programming language, a reasonable learning objective might be to teach students to program. However, to achieve this objective, an efficient pedagogical approach must be chosen. Lemos suggests good approaches might include structured programming, modular programming, grammatical or whole problem approaches, a spiral approach, a problem analysis approach, computer-assisted instruction, instructional television, egoless[5] programming or team debugging. It seems evident, however, that in order to determine the most effective approach (or combination of approaches) the learner's particular cognitive style and consequent learning strategies should be taken into account.

Cheney (1980) describes a study conducted to examine the relationship between cognitive style and student programming ability. He concludes that analytic, field inde-

5 In which a team is responsible for programming and where no one individual assumes final responsibility.

pendent thinkers will be more successful than heuristic thinkers. As this is a cognitive style displayed by successful language learners, and as programming languages are similar to natural languages in many respects, it seems possible to predict the successful programmers using a battery of instruments shown to be effective in predicting successful language learners.

Computer Literacy

A. Definition

The search for a comprehensive definition of computer literacy encompassing aspects from awareness to programming continues. It seems evident that computer literacy should include more than just programming or a general awareness of how computers are used every day. Watt (1981) bases a rather comprehensive definition of computer literacy on his interpretation of the common meaning of the word "literacy", which he sees "as a continuum from a minimal ability to read news headlines to the literary skills of a professional writer or skilled academic at the other". As Watt's definition has implications for the basic premise of this study, that is that computer literacy should be available to all students at all levels and of all abilities; it seems worthwhile to examine it in some detail here.

Watt divides his definition into four areas. Computer Literacy is:

1. The ability to control and program a computer to achieve a variety of personal, academic, and professional goals. This includes the abilities to read, understand and modify existing computer programs and to determine whether or not the program and/or the data it is using are correct and reliable.

2. The ability to use a variety of programmed computer applications in personal, academic and professional contexts.

3. The ability to understand the growing economic, social, and psychological impact of computers on individuals, on groups within our society, and on society as a whole.

4. The ability to make use of ideas from the world of computer programming and computer applications as part of an individual's collection of strategies for information retrieval, communication, and problem solving.

Ragsdale (1982) describes a series of "eras" of computer literacy which overlap to some extent. After the first two eras in which information about computers was dispensed at the graduate and undergraduate levels at universities, secondary schools in the third era will become responsible for imparting computer skills to increasingly sophisticated students and will gradually, in the fourth era pass this responsibility down to the elementary schools. This implies that in the near future all students will, upon entering secondary schools, be able to operate microcomputers, be familiar with their applications, and generally be able to write some programs. In particular, this suggests that well-defined and well-planned programs of computer literacy

need to be in place at the secondary level in the very near future, based on what is known about the strengths and needs of the students involved.

At the present time, the average citizen remains _culturally disadvantaged_ vis à vis computers and the intrinsic part they play in his or her life (Goddard, W.P. and Wright, A.E.). This "ignorance is bliss" attitude will improve far too slowly if the following statements continue to apply to our schools (Goddard, W.P. and Wright, A.E., p. 3).

> 1. There are only enough teachers and equipment to provide special computer courses at the upper high scool level.
>
> 2. School computers are generally considered to be the province of the high school mathematics department.
>
> 3. A mythology has developed around computers - they are unfamiliar, difficult, and require a special kind of brain power.
>
> 4. Applications to areas other than mathematics has been limited.
>
> 5. Because information ... is not made available to all subject area teachers, there has been little opportunity to be creative in the use of computers in schools.

Certain authors suggest that schools which fail to provide the opportunity for pupils to develop an understanding of the capabilities and limitations of the computer in society are remiss in fulfilling their mandated responsi-

bilities (Hull, T., Holt, R.C., and Phillips, C., 1975; and Johnson, D.C., Anderson, R.E., Hansen, T.P., and Klassen, D.L., 1981).

B. Curriculum

While there is general agreement as to the concepts to be included, there is as yet no cohesive nation-wide program of computer literacy available. Thus, school divisions across the continent have endeavoured to develop their own curricula based on carefully defined notions of computer literacy. Two such efforts are described below.

The Cupertino-Union School District (California) drafted an original curriculum in 1981, which was revised in 1982. Defining computer literacy as "the ability to function in a computer and technology oriented society ... (to) understand computers and their applications and implications ... (and to) develop the skills necessary to communicate with computers and recognize the computer's capabilities and limitations" (p. 7), the Committee members established a comprehensive curriculum that recognized that, given time, most children could achieve all objectives identified. This curriculum is divided into three main areas of literacy; awareness, interaction, and programming, and objectives are indicated for each area as well as for the core subject areas (Math and Science being only two). An interesting (and heartening) feature of this curriculum is its impartiality as regards subject area and ability level. All students are expected to achieve the basic objectives on

some level of competence regardless of their strength (or lack of it) in Math/Science. The objectives are listed in Appendix B of this thesis.

The Albany School District plan, very similar to the Province of Alberta plan, divides the curriculum into three levels with specific goals for each level. Whereas the Cupertino plan suggests a "spiral" approach, the Albany plan is very specific in the separation of its broad goals (use, programming, instructional uses, parts and functions, and vocational uses and impact) into elementary (K - 5), middle school (6 - 8) and high school level (9 - 12) courses. The bulk of programming instruction is left to high school elective programs, although it is estimated that 30% of students will have had <u>some</u> instruction in programming by the time they reach this level. The "Computer Use Framework" is included in Appendix B.

C. <u>Programming Instruction</u>

Common to all definitions of computer literacy is the concept of "programming". There is some confusion and disagreement, however, as to what "programming" implies, especially at the secondary level.

Ragsdale (1981) and Moursund (1973) both point out that programming initially involved graduate, then undergraduate university level students who learned, for the most part, FORTRAN. Programming instruction has gradually "filtered" down to the secondary schools as more students at that level

enter their classrooms already somewhat familiar with computers and their uses.

The questions "How much programming?", "What language is best?", and "When to start?" have yet to be resolved to anyone's satisfaction, although certain "experts" in the field have endeavoured to present general guidelines.

Moursund (1973) lists nine possible goals for a course in introductory programming at the secondary level, noting that a typical course would not involve <u>all</u> the goals:

1. To teach problem analysis from a computer-oriented point of view.
2. To teach a subset of a compiler language (such as BASIC ...) and to give the student skill in programming in the chosen language.
3. To present the capabilities and limitations of computers (or language), and implications of the ready availability of computers.
4. To acquaint the student with the idea of machine and assembler language programming.
5. To teach the student how to use packaged ("canned") programs.
6. To give the student sufficient training in computing so that he can communicated with a programmer.
7. To introduce the student to the basic concepts, ideas and goals of computer science.
8. To instill in the student certain attitudes toward computers.
9. To provide a combination of the above points which will prepare the student to go on to a more advanced course in computer science.

These nine general goals have a close affinity to the goals of modern language teachers and students. Goals 1 and

2 deal with establishing a minimum communicative competence in the target language with a view to using it in "real" communication situations. Goal 6 establishes the desirability of developing contact with a "native" speaker, a strategy used by the Good Language Learner. Similarly, Goal 8 is based on instilling empathy for and a positive attitude towards the language and its speakers, while Goal 7 involves the teaching of "culture". It is, obviously the ultimate goal of all language programs to encourage students to progress to more advanced levels.

Ragsdale (1981) extends this analogy to second language learning in discussing Ershov's equation of computer literacy with computer programming (p. 17). Ragsdale argues that generally we use reading as a primary tool for instruction, with writing being used primarily in evaluation of learner achievement. Particularly in dealing with a second language, the mastery of the language is enhanced through the reading of poems, songs, stories, etc., while the writing of text seems less crucial.

> Similarly, one could argue that reading or understanding computer programs is more important to the attainment of a general level of computer literacy, though some program writing would be useful. The rationale for this belief is that writing in any language, including computer languages, requires greater mastery of the syntax rules which ... are often complex ..." (p. 17).

Based on the above argument, it seems reasonable to teach young (elementary) students to read and understand computer programs, while including a limited amount of pro-

gram writing, especially for those students who seem to "require" advanced instruction. Ragsdale suggests that since the schools might benefit from student programmers (a possible benefit might include peer tutoring), who themselves will benefit from programming experience in demonstrating their understanding of other subjects (p. 18), instruction in computer program reading should be a component of computer literacy before grade six. An emphasis on computer _programming_ would then be appropriate at the secondary level.

In order that all students be encouraged to pursue to some degree competency in programming, this aspect of computer literacy needs to be very carefully planned. Because any acceptable and adequate program of literacy in any subject area must include a remedial component, it would be short-sighted to neglect this aspect in planning a program of computer literacy. For this reason, it is essential to identify early those students who will not be successful in a program based on the cognitive styles and learning strategies of the _good_ language learner.

CHAPTER III

Design of the Study

Selection of Test Instruments

It was proposed in Chapter II that similarities between natural and programming languages suggest that instruments used to predict successful natural language learners could conceivably be used to predict successful programmers. Based on the study conducted for OISE by Naiman, et al. (1978), also referred to in Chapter II, in which the Good Language Learner was profiled, two test instruments suggested themselves to this researcher as being especially appropriate: _Budner's Scale of Tolerance/Intolerance of Ambiguity_ and the _Group Embedded Figures Test_. The characteristics measured by these two instruments correlated more highly to scores on the International Education Association Test of French Achievement than the other characteristics measured.

A. _Budner's Scale of Tolerance/Intolerance of Ambiguity_

The degree to which an individual is willing to tolerate ideas or systems contrary to his/her own, as measured by _Budner's Scale of Tolerance/Intolerance of Ambiguity_, is agreed to be an important factor in successful second language learning (Naiman, N. et. al, 1978). A learner with a high degree of tolerance is less likely to react to the

[illegible]

foreign language situation in negative ways that subvert the learning process.

Budner (1962) defines intolerance of ambiguity as "the tendency to perceive ambiguous situations as sources of threat" while tolerance implies "the tendency to perceive ambiguous situations as desirable" (p. 29). He offers the following examples of situations that might be classed as ambiguous, or as situations which are difficult to categorize because of the lack of sufficient clues:

1) A completely new situation containing <u>no</u> familiar clues.

2) A complex situation containing <u>many</u> clues.

3) A contradictory situation in which different elements or clues suggest different structures. The learning of a new language could conceivably fall into all three categories.

He further classifies reactions to ambiguous situations into two categories: Denial and submission.

Submission, defined as the recognition of the situation as an unalterable fact of existence, may take two paths: Phenomenological (anxiety, discomfort) and operative/avoidance behaviour.

Denial, which is the performance of an act by which the reality of the situation, even if only in the phenomenological world of the individual, is altered to suit his/her desires, is likewise subdivided into two categories - that of phenomenological (repressed, denial) and operative (destructive or reconstructive) denial.

Derived primarily from Frenkel-Brunsurck's work in attempting to establish a relationship between this variable and the authoritarian syndrome (2a), Budner developed a 16-item scale sampling specific types of behaviour (see Appendix A).

When used as a cognitive style indicator in the OISE study (Naiman et al.) the scale was in a 16-item format rated by a 7-point Likert scale. This measure was found to correlate highly with the International Education Association tests of French achievement.

B. <u>Group Embedded Figures Test</u>

Cheney (1980) describes a study investigating the relationship between cognitive style and success in learning a programming language. The study involved undergraduate students learning BASIC. The students' cognitive style was considered on a continuum between totally analytic and totally heuristic (Haysman, J.H., 1970, and Lusk, J. and Kersnick, 1979).

Analytic reasoning, related to field independence, involves a structured approach to problem solving. Often, all effort is directed towards detecting underlying relationships and attempting to reach an "optimal equilibrium".

Heuristic, or global reasoning, frequently emphasizes workable solutions to the total problem situation. Typically, heuristic reasoning involves common sense, intuition, and a search for analogies with familiar solved problems.

Obviously, there is a continuum of "ideal" types, although Cheney found that primarily analytic thinkers tended to be more successful in learning BASIC as it was taught at a particular university. Likewise, Naiman and others and Pimsleur (1962) found that good language learners tended to be field independent. Consequently, the _Group Embedded Figures Test_ (GEFT) developed by Witkins et al. (1971) was administered to the group of junior high school students participating in this study to determine where each student falls along the continuum.

An adaptation of the _Embedded Figures Test_ (EFT), for large group administration contains 18 complex figures, 17 of which were taken from the EFT. Shading serves to embed simple forms - the subject is prevented from seeing simultaneously a simple form and the complex form containing it.

The test is composed of three parts:

1) Part I contains seven very simple items and serves as practice.

2) Parts II and III each contain nine more difficult items. The test is timed and it is scored on the basis of the number of questions completed correctly. In order to be correct, _all_ lines must be traced, with none added.

Table II presents the norms and reliability coefficients for the GEFT, based on undergraduate men and women at an eastern university in the United States.

Table 2. Norm and Reliability Coefficients for the GEFT

Quartile	Number Correct on the GEFT	
	Men	Women
1	0-9	0-8
2	10-12	9-11
3	13-15	12-14
4	16-18	15-18
N	155	242
Mean	12.0	10.8
SD	4.1	4.2

In addition, the GEFT was compared to other cognitive style measures such as the EFT, yielding the following correlations between the GEFT and the EFT:

.82 for men

.63 for women

With high correlations a strong case can be made for the GEFT on the basis of convenience as the EFT must be administered individually, each session taking 15 to 20 minutes.

C. The Modern Language Aptitude Test (MLAT)

Originally, the MLAT was included in this study as a measure of aptitude that had a high degree of validity when

employed with the age group recommended. However, no form of the MLAT for Grade 7 students exists. (The EMLAT is intended for elementary students to Grade 6, while the MLAT is for students Grade 9 to adult.)

Since the MLAT tests skills measured by other instruments, it was (regretfully) omitted from this study.

Selection of the Students

In January 1982 a number of junior high school teachers interested in the programming aspect of computer literacy were approached regarding this study. A junior high school in southeast Edmonton that was willing to provide access to its entire Grade 7 population was finally selected. These Grade 7 students had access to a "Computer Club" which was made available at noon and after school, during which times five Apple II Plus microcomputers were available under teacher supervision. A Computer Literacy Option had been offered at this school during the year but Grade 8 and 9 students had had priority, so few Grade 7 students had been enrolled. In keeping with the delimitations of this study (see Chapter I), only those students who had not been involved in this option were included in the study (N=158).

Access to the School

In consultation with the principal of the school and the Grade 7 teachers involved, a time-line for the study was

established. Those students involved in this study would participate in the option in lieu of one forty-minute health period per week for approximately eight weeks. The study would take place at the same time as the other B-option courses were offered in the school. Consequently, one of the classes of Grade 7 students was a composite of students from the other classes that had indicated Computer Literacy as their first choice on the list of available B-options, but had not yet taken such a course.

The Procedure

A. Pilot Testing

The Budner Scale of Intolerance of Ambiguity was administered to 24 students chosen at random from among the six participating classes. This pilot was intended to identify any problems with reading comprehension on the test. Prior to this first session, the scale was discussed with two language arts teachers on staff at the school, both of whom felt that the statements were at a reading level beyond the average Grade 7 student. During the forty-minute period in which the scale was administered, the students were encouraged to comment on the wording of the statements and to note any uncertainty or incomprehension. As each student completed the scale, he/she was questioned individually by the researcher about the vocabulary and structure of the statements. Based on this questioning, it was decided to

[illegible]

eliminate the scale from the study as rewording the items would affect the validity of the instrument.

B. Testing

Over the course of the first week in the school all Grade 7 students participating in the study completed the Group Embedded Figures Test.

The following week these students began a short programming course organized on a B-option basis. This programming course is described in detail in the next section of this chapter.

Upon completion of the course, the students were graded on a 3-point system (E = Excellent; S = Satisfactory; N = Needs Improvement) after meeting certain criteria in a completed computer program, and after interpreting a short program listing. These criteria are discussed in the next section of this chapter.

Teaching Computer Programming

A group of 158 Grade 7 students participated in this "option" beginning April 25, 1983 and continuing until June 16, a date made necessary because of the impending final exams. As a result, the researcher was present in the school and actually teaching for eight weeks, so that each class (of seven total classes) received 360 minutes of instruction and practice in programming in BASIC.

A. Nature of the Sessions

Each instructional session was organized into two parts: Demonstration of a programming concept at the beginning of the session, lasting about ten minutes; followed by "hands on" practice on the computer for the remainder of the forty minutes. As the class ranged in size from 15 to 30, each student had limited practice time, although the students were encouraged to use time not actually spent on the computer in planning. The computers were also made available at noon and after school on the days the researcher was present in the school.

At the beginning of each session a summary of the preceding lesson, with accompanying problems, was discussed. These are included in Appendix C.

As a result of discussions with Dr. Tom Kieren of the Faculty of Education, University of Alberta, with regard to the assessment of computer programming competency, the following criteria were adopted:

1) <u>Completion</u>

   The student will complete a skeleton program including syntax and semantics.

2) <u>Organization</u>

   Given a problem, the student will develop a plan, and will analyze a problem without seeing it run.

3) <u>Logic, Critical Command</u>

The student will be able to write a program to solve a problem using a specific code.

4) <u>Application</u>

The student will develop a small program illustrating particular concepts.

Achieving all four of the above objectives would yield a score of Excellent (3), while mastery of three out of four objectives would yield a score of Satisfactory (2). Mastery of less than three objectives yielded a score of Needs Improvement (1). In order to cover all of the objectives and in order to provide an opportunity to both read and write a program, each student participating in the option completed two programming exercises. Programming 1 (P1) involved writing a program and saving it on a disk, while Programming 2 (P2) involved analyzing a program listing without seeing it run. The latter exercise is included in Appendix C.

The program writing took place over the last three weeks of the course. Each student had approximately forty minutes to write a short program that:

a) would run without errors;

b) included the VTAB and HTAB commands and was properly formatted;

c) included a FOR-NEXT loop or an example of low resolution graphics;

for a rating of Excellent (3) or:

a) would run without errors;

b) was properly formatted using PRINT statements;

for a rating of Satisfactory (2). Programs that included either syntax or formatting errors or that were incomplete received a score of (1), or Needs Improvement. The Programming 2 exercise was completed separately during another class.

Analysis of the Data

1) A multiple linear regression analyis was performed on the following data:

   a) Total scores on the GEFT.

   b) Data obtained from cumulative records which included verbal, non-verbal, and quantitative IQ scores (Canadian Cognitive Abilities); reading comprehension and decoding scores (Reading Survey Test, E.P.S.B.).

   c) Teacher-assigned percentage grade scores in the academic subjects for the previous term's work (language, arts, math/science, social studies).

   d) Teacher-assigned percentage grade scores in French language arts.[6]

6 Measures (3) & (4) above were obtained from the school's academic records and reflect an averaging of test scores, writing assignments, oral competence (in French) and participation in class.

e) Registration (or not) in a French option or French immersion.

f) Sex, age and class.

g) Success in programming indicated by a 3-point scale (E-S-N) as measured by a two-part test.

2) Multiple correlation coefficients were determined among all pairs of variables to indicate the factors that were significant ($p \leq .05$) in predicting students most likely to succeed in learning a programming language as taught by the researcher.

3) For the 3 categories (E,S,N) the mean, media and quartile distributions were found for all the variables.

CHAPTER IV

RESULTS OF THE STUDY

The focus of this study was to test selected variables as predictors of success in computer programming at the secondary level. It was hoped that the results of this analysis would make it possible to identify prior to the computer programming component, students who potentially might face particular problems in order that they could be given special assistance to ensure them success.

An examination of the data showed that, of the total 16 predictor variables available, 11 qualified for consideration for P1[7] and only 6 qualified for P2. This difference in number of predictor variables is consistent with Ragsdale's observation, mentioned in Chapter II, that writing a program requires greater mastery of the rules of syntax than reading one (Ragsdale, 1981). If this is so, one would expect more variables to be relevant.

The variables' correlations with either of the two measures of programming success, their order of entry into the prediction equation and the multiple correlation (R) with criterion are shown in the following tables.

As Table III shows, of the 16 predictor variables for P1, 11 were significant, 3 at the .05 level, and 8 at the

7 P1 involved writing a short program (to meet criteria described on pages 39-40) and saving it to a disk. P2 involved interpreting a written program.

.01 level. The positive correlation of success in P1 between score on the Group Embedded Figures Test supports the findings of Cheney (1980) who found that there was a high

Table III. Correlation Co-Efficients of Sixteen Predictor Variables With Each of Two Criterion Scores N=158.

Predictor	Criterion	
	P1	P2
Class	0.0081	0.2765
Sex	-0.0677	-0.0436
Age (months)	0.0161	-0.0943
IQ - Verbal	*0.1879	0.1290
IQ - Quantitative	**0.2609	*0.2942
IQ - Non-Verbal	**0.3263	**0.3483
Reading - Decoding	*0.1729	*0.1557
Reading - Comprehension	*0.1829	0.0764
Language Arts	**0.3015	0.1121
Math/Science	**0.2626	0.1420
Social Studies	**0.2278	0.0121
French Enrollment	**0.2430	0.0843
French Score	0.0838	0.0845
GEFT	**0.3835	**0.3122
P1	1.0000	**0.4250
P2	**0.4250	1.0000

* p < .05

** p < .01

Table IV. Summary of Means, Medians and Standard Deviations For Category 1 of P1.

Variable	$\overline{X}$	M	SD
Class	3.625	3.955	1.561
Sex	1.438	1.389	0.504
Age (months)	149.032	147.300	7.445
IQ - Verbal	107.517	106.000	13.540
IQ - Quantitative	103.207	102.333	12.985
IQ - Non-Verbal	101.517	106.000	12.540
Reading - Decoding	30.231	31.000	6.244
Reading - Comprehension	69.269	69.500	18.096
Language Arts	58.188	54.500	14.261
Math/Science	62.125	62.000	17.843
Social Studies	58.656	59.000	16.746
French Enrollment	0.281	0.196	0.457
French Score	3.556	3.200	1.130
GEFT	6.500	4.833	4.478

Table V. Summary of Means, Medians and Standard Deviations for Category 2 of P1.

Variable	$\overline{X}$	M	SD
Class	3.402	3.192	1.858
Sex	1.573	1.628	0.498
Age (months)	148.800	147.231	7.041
IQ - Verbal	110.467	112.250	14.212
IQ - Quantitative	103.853	105.750	13.311
IQ - Non-Verbal	109.919	111.500	13.760
Reading - Decoding	31.423	32.143	5.749
Reading - Comprehension	74.493	76.750	14.884
Language Arts	65.481	68.125	14.397
Math/Science	68.190	70.583	13.967
Social Studies	61.949	62.000	16.093
French Enrollment	0.658	0.740	0.477
French Score	3.245	3.583	1.479
GEFT	9.557	9.286	4.466

Table VI. Summary of Means, Medians and Standard Deviations for Category 3 of P1.

Variable	$\overline{X}$	M	SD
Class	3.649	3.417	1.367
Sex	1.351	1.271	0.484
Age (months)	149.333	148.286	6.693
IQ - Verbal	116.059	118.500	19.091
IQ - Quantitative	113.235	111.167	12.063
IQ - Non-Verbal	115.324	116.000	11.028
Reading - Decoding	33.182	34.875	5.468
Reading - Comprehension	77.606	80.000	12.150
Language Arts	71.595	76.333	15.126
Math/Science	74.568	80.000	17.438
Social Studies	70.027	77.000	18.908
French Enrollment	0.649	0.729	0.484
French Score	3.708	3.944	1.233
GEFT	11.972	12.000	4.411

correlation between an analytic cognitive style and a student's ability to write computer programs. This would seem to explain the positive correlation between P1 and achievement in math and science as the analytic cognitive style is associated with success in math and science. It seems likely that the positive correlation between P1 and language arts and social studies scores is related to the ability to read and analyze materials related to problem-solving. All three measures of IQ related positively to success in P1; similarly enrollment in French (either as a second language or immersion) was a predictor of success in programming. This latter seems reasonable in the sense that the academically-inclined students tend to choose French as an option at the secondary level; however, it may also indicate that students enrolled in prior language learning programs have better opportunity to develop the cognitive styles applicable to learning a programming language.

Somewhat surprising at first was the correlation between P2 and Class. Upon reflection, this result was sensible since Class 1 was the first "run-through" of that week's lesson, whereas the last class was not taught this material until Friday.

It would seem, from comparing the means, medians and standard deviations for each of the predictor variables for the three categories of achievement in P1 that the following generalizations might be made. The successful (category 3 =

Excellent) programmer involved in this study would be twelve years old (as opposed to a student who had entered Grade 1 at seven years, or who had repeated a grade), with mean scores of 116, 113 and 115 on IQV, IQQ and IQN, respectively, 33 on RD and 77 on RC, and school grades ranging in the low to mid seventies in academic subjects. This student would be a French student achieving honors, and an analytic thinker. These results seem to support McEwan (1973) who found that successful FSL students were those who tended to achieve high grades in all academic subjects, and Naiman et al. (1978) who found that good language learners tended to be analytic as opposed to heuristic thinkers. If we accept that learning a programming language requires many of the same strategies as learning a "natural" second language, such as French, then these results would seem reasonable.

Table VII provides a summary of the quartile distributions for significant variables for P1. Each variable is discussed below.

IQ - Verbal

The lowest achievers (1 = N) in P1 had IQ - Verbal scores ranging between 80 - 126, whereas average (2 = S) achievers ranged from 74 - 142 and high achievers (3 = E) ranged from 67 - 144. High achievers did not have significantly higher scores on IQV than average achievers.

Table VII. Summary of Quartile Distributions for Significant Variables for Three Categories of P1.

Quartile	Category 1				Category 2				Category 3			
Variable	1	2	3	4	1	2	3	4	1	2	3	4
IQV	80-96	100-106	107-122	123-126	74-101	103-112	113-120	121-142	67-103	105-118	121-130	132-144
IQQ	79-93	98-102	103-111	112-133	68- 95	96-106	108-113	115-124	92-104	105-111	113-119	120-139
IQN	65-88	92-101	106-116	117-122	74-101	103-111	112-118	119-143	88-109	110-116	117-121	122-143
RD	18-24	27-31	32- 35	38-39	11-27	29-32	33- 35	36-42	18-29	31-34	35-37	38-42
RC	31-58	59-69	73-85	87-93	37-66	67-76	77-86	87-95	51-72	74-80	81-85	86-94
LA	37-46	48-54	55-67	90-93	0-56	57-68	69-76	77-91	27-58	62-76	78-80	81-90
MS	32-45	48-62	63-77	80-93	27-60	62-70	71-78	80-96	31-62	63-80	81-87	88-96
SS	32-42	44-59	60-72	75-95	27-50	51-62	63-73	74-96	28-57	61-77	78-83	86-92
F Score	-	-	-	1	-	-	-	1	-	1	1	1
GEFT	1-3	4-5	7-9	10-16	1-2	7-9	10-13	14-18	0-8	9-11	12-15	16-18

According to Thorndike and Hagen (1974) students who score high on the Verbal and Quantitative Batteries have "well-developed analytic reasoning skills, high levels of abstract reasoning, ... and tend to be very resistant to distraction". If we study Table XI, we note that IQV had a high positive correlation with IQQ. However, each measure correlates positively with the GEFT at the $p < .01$ level. Noting that students who score highly on either IQV or IQQ tend to do well where the pace of instruction is relatively fast (as in a programming or second language class), requiring the students to make logical inferences quickly and accurately, the authors of the Canadian Cognitive Abilities Test (CCAT) nevertheless note that students scoring lower on these two measures may still be successful where the instructional material is carefully controlled over a long period of time for both instruction and practice purposes.

One would expect that students who scored lower on the IQQ test would have less trouble learning a programming language than those scoring lower on the IQV measure. Learning BASIC at this level requires a reasonable degree of competence in solving mathematic-related problems. This is borne out by the higher correlation of P1 with LA than MS.

IQ - Non-Verbal

Scores on this measure ranged from 65 - 122 for low achievers, from 74 - 143 for average achievers, and from

88 - 143 for high achievers. This measure also correlated more highly than either IQV or IQQ with success on P1 and P2. Thorndike and Hagen (1974) indicate that students who score higher on this battery usually have well-developed reasoning abilities but process information quite differently from the high-verbal student. Apparently, these students are "exceptional in perceiving and manipulating spatial relationships, in generating a concept of the whole from fragmentary information about the parts, in discerning patterns, and in perceiving and remembering stimuli that either have no verbal label or are too complex to specify in words". Children with rich visual imagery would seem fated to be particularly intrigued with and successful at programming.

The argument that higher scores on either IQV or IQN are more significant in predicting success in programming is supported by both the multiple regression analysis, on which IQN entered on step 4 for P1 and step 1 for P2; and IQV entered on step 5 for P1. Evidently, a higher score on the IQN battery is a better predictor overall than either of the other two IQ batteries.

IQ - Quantitative

Scores on this measure ranged from 79 - 133 for low achievers on P1, from 68 - 124 for average achievers, and from 92 - 139 for high achievers.

Reading - Decoding

Scores on this measure ranged from 18-39 for low achievers, 11-42 for average achievers, and from 18-42 for high achievers on P1. Higher scores in the fourth quartile are remarkably similar, which is also the case for scores on RC (31-93, 37-95, 51-94). These results would seem to bear out the conclusions of Lemos (1980) that a "moderate relationship appears to exist between reading and writing ability" in computer programming. Although he was referring to the ability to read and analyze computer programs, it would seem logical to expect "good" readers of programs to possess the same skills as "good" readers of prose; namely sensitivity to grammatical construction, extensive vocabularies, and ability to make inferences and predictions based on what is read. Lemos suggests further that reading tests with a high degree of relationship to program writing capability would be a powerful tool for instructors. As Reading - Decoding, in particular, shows a positive correlation with both P1 and P2, it could provide a good starting point for developing such measures.

Language Arts

Language Arts scores showed the third highest positive correlation with P1 after the GEFT and IQN. Students falling into the fourth quartile in all three categories of P1 achieved over 75% in this academic subject. A more detailed

analysis of the data yields the highly significant correlations of 0.4739 with IQN, 0.5105 with RD, and 0.5476 with RC. It seems that reading ability is a significant factor in success in programming ability.

Math/Science

Although achievement in Math/Science (M/S) seems less important than achievement in LA, it is nonetheless significant at the $p < 0.01$ level. Similarly, quartile distributions ranged from 27 to 96 in all three categories, with the fourth quartile in all three categories ranging from 80 or above to 96. It seems that students with MS ability will probably be successful in a programming option, although a better academic predictor would be achievement in LA.

Social Studies

Again, high achievers in Social Studies obtained scores from 74 to 95 in the fourth quartile in all three categories on P1. Generally, the high academic achievers seem more likely to be successful in exercises of programming ability.

French Enrollment

Interestingly enough, enrollment in French was more significant than actual achievement in French. On further reflection, this is not surprising as the academically-inclined students tend to choose French as an option in

junior high school. These students then fall into levels of achievement among themselves while remaining the "top" students in the school. It should be noted here that the group of Grade 7 students at Kenilworth Junior High School included a class of French immersion students (N=15). Incidently, LA correlated positively (0.4422) with French Score.

GEFT

Finally, the quartile distributions for the GEFT, which had the highest positive correlation with P1 in the study, are represented in Table VII.

The first quartile scores in each category began at 0. Not surprising is that the highest score in the first quartile in Category 3 (8) is higher than the highest score in the second quartile of Category 1 or similar to the range of scores in the second quartile in Category 2. In other words, the lowest scores on the GEFT in Category 3 of P1 are still higher than half the scores in the other two categories. This indicates a strong relationship between analytic thinking and success in programming.

Table VIII. Correlations Between Predictors and P1 and Results of Multiple Regression Analysis.

Predictor Variables	Correlation With P1	Order of Entry Into Regression Equation	Resulting _R_
GEFT	0.3835**	(1)	0.378
LA	0.3015**	(2)	0.428
Sex	-0.0677	(3)	0.439
IQN	0.3263**	(4)	0.445
IQV	0.1879*	(5)	0.452

* $p \leq 0.05$

** $p \leq 0.01$

Table IX. Correlations Between Predictors and P2 and Results of Multiple Regresion Analysis.

Predictor Variables	Correlation With P2	Order of Entry Into Regression Equation	Resulting $\underline{R}$
IQN	0.3483**	(1)	0.307
RC	0.0764	(2)	0.344
GEFT	0.3122**	(3)	0.358

* $p \leq 0.05$

** $p \leq 0.01$

Table X. Comparison of Correlations and Results of Multiple Regression Analysis Between Predictors for P1 and P2.

Predictor Variables	Correlation With P1	Correlation With P2	Order of Entry Into Regression Equation		Resulting $\underline{R}$
			P1	P2	
Sex	-0.0677	-0.0436	(3)	-	0.439
IQV	0.1879	0.1290	(5)	-	0.452
IQN	0.3263**	0.3483**	(4)	(1)	0.307
RC	0.1829*	0.0764	-	(2)	-
LA	0.3015**	0.1121	(2)	-	0.428
GEFT	0.3835**	0.3122**	(1)	(3)	0.358

* $p \leq 0.05$

** $p \leq 0.01$

Table X. Comparison of Correlations and Results of Multiple Regression Analysis Between Predictors for P1 and P2.

Predictor Variables	Correlation With P1	Correlation With P2	Order of Entry Into Regression Equation		Resulting $\underline{R}$
			P1	P2	
Sex	-0.0677	-0.0436	(3)	-	0.439
IQV	0.1879	0.1290	(5)	-	0.452
IQN	0.3263**	0.3483**	(4)	(1)	0.307
RC	0.1829*	0.0764	-	(2)	-
LA	0.3015**	0.1121	(2)	-	0.428
GEFT	0.3835**	0.3122**	(1)	(3)	0.358

* $p \leq 0.05$

** $p \leq 0.01$

The variables which were correlated at at least the .05 level of significance and those which entered the regression equation totalled five for P1 and three for P2. Common to both measures of programming ability were the following variables: _IQN_, _RC_ and the _GEFT_. All three of these variables seem to act as suppressors on the other predictor variables which entered into the equation. That is, each variable has a low correlation with the criterion (P1, P2) and a high correlation with one of the other predictor variables. Table XI exposes the inter-correlations of all of the predictor variables in the study.

For example, looking first at _RD_ to determine why it didn't enter into the equation, we find it was highly correlated with _IQN_ (r = .6356) which entered the equation for P2 on step 1. As a result, it might have accounted for some of the variance in the criterion as IQN and, therefore, its predictive value was lowered when IQN entered the equation. We could account for the non-entry of _MS_ and _IQV_ in the same way.

In comparing the results of the analysis for P1 and P2, a number of differences become apparent. First, only two of the variables were common to both equations: _IQN_ and _GEFT_. Notable by its absence was _LA_ for P2 which entered the equation for P1 on step 2; and _RC_ for P1 which entered the equation for P2 on step 2. Since these two variables correlate highly (r = .5476), we might account for the difference in this way.

Table XI. Inter-Correlation Matrix.

Variable Names	Class	P/C	Sex	Age	IQV	IQQ	IQN	RD	RC	LA	MS	S	F	FS	GEFT	P1	P2
P2	.2765	.0212	-.0436	-.0943	.1290	.2042	.3483	.1557	.0764	.1121	.1420	.0121	.0843	.0845	** .3122	** .425	1.000
P1	.0081	-.0058	-.0677	.0161	.1879	.2609	.3263	.1729	.1829	** .3045	** .2626	** .2278	** .2430	.0838	** .3835	1.000	
GEFT	.0490	.0187	-.0256	-.1205	** .3023	** .4500	** .5702	** .3725	** .3239	** .3694	** .4470	** .2296	** .3312	.1576	1.000		
FS	-.2021	-.0178	-.0837	.1676	.1800	* .2584	* .3425	** .3101	* .2545	** .4422	** .3944	** .3737	* .1873	1.000			
F	-.0790	-.0869	.1943	* -.1432	** .3834	** .3033	** .3685	** .3817	** .4140	** .5183	** .3348	** .2946	1.000				
S	-.0567	-.0106	.1155	-.1026	** .4250	** .4854	** .3189	** .3570	** .4238	** .6896	** .6918	1.000					
MS	-.0286	-.0392	.0045	* -.1789	** .4595	** .6263	** .4898	** .4420	** .4801	** .7364	1.000						
LA	.0214	-.0438	* .1657	** .2375	** .5923	** .5987	** .4739	** .5105	** .5476	1.000							
RC	-.0443	* -.1718	.1076	** -.2622	** .7651	** .6257	** .6266	** .8156	1.000								
RD	-.0034	-.1020	.7323	** -.4092	** .7773	** .6503	** .6356	1.000									
IQN	-.0036	* -.1533	.0305	** -.2901	** .6044	** .6402	1.000										
IQQ	-.0996	-.0855	-.0339	** -.3571	** .6420	1.000											
IQV	.0146	-.1184	.0885	** -.3859	1.000												
Age	.0070	* .1491	-.1267	1.000													
Sex	-.0271	.0084	1.000														
P/C	.0644	1.000															
Class	1.000																

* p < .05

** p < .01

However, as P2 required the reading and analysis of a program as opposed to the writing of a program, it seems reasonable that reading ability would have a greater influence on a measure of this sort than <u>LA</u> which encompasses all the skills associated with language.

CHAPTER V

CONCLUSIONS AND DISCUSSION

Summation

Johnson and others, in "Computer Literacy and Awareness" (1981) asks "What should the educated citizen know to be able to function as a contributing member of a society which utilizes computers in the home, business, industry and government?". In education we tend to suggest that while there may be some 'minimal level' of understanding, it is also the goal that each child should be educated to his or her full potential. To be computer literate implies comprehension and the ability to discuss computing concepts, applications and issues intelligently.

> It should be the goal of education to provide an opportunity for all pupils to reach a level of literacy commensurate with each individual's potential. (p 15)

The latter sums up quite succinctly the basis for doing a study of this nature. Earlier it was stated that results arising from such an inquiry could be used as a guideline in helping to predict not only success in learning computer programming, but more importantly would suggest possibilities for teaching techniques more effective for "weak" students. By being aware of the particular learning style(s) and their relationship to the learning of programming lan-

guages displayed by students who were unsuccessful in _this_ study, the computer literacy program for junior high should be revised at some point to provide successful experiences for all.

The original question asked in this study was: What are those factors that will allow us to predict success in computer programming at the junior high school level in Alberta? The results of the multiple correlation and multiple regression analyses indicate that the successful student will be an analytic (as opposed to heuristic) thinker, scoring high on the GEFT; will be in possession of a "rich visual imagery", scoring high on the IQN battery; will tend to have high verbal or abstract reasoning skills; will do well in the academic subjects, especially LA; and will be a enrolled in French where such a program is available. Slightly less important as a predictor, seemingly, is actual performance in French. This is not surprising to any junior high school teacher, who knows that the academic students tend to choose French as an option so that those students fall into categories of achievement _within_ that group. It seems reasonable to conclude that, in general, those students identified as high academic achievers will be successful in a computer literacy course _taught as in this study_, with a strong emphasis on programming skills.

Discussion of the Study

Based on the researcher's experience in teaching computer programming at this level, the following changes and recommendations should be considered:

1. With such a short period of time in which to teach an option of this type, I should have concentrated on low-resolution graphics instead of mastery of such "complex" notions such as FOR-NEXT loops. Graphics are easily learned, and highly motivating in that "something happens" on the screen immediately, more students have the opportunity to demonstrate their creativity, and such programs are relatively easy to debug.

2. As "hands-on" time was very limited, more use should be made of peer tutors in other grades who could conceivably receive credit for their involvement.

3. More time should have been spent in planning. I should have insisted that <u>no</u> <u>one</u> touch a keyboard without a plan on paper, thus eliminating sitting blankly in front of a flashing cursor while others were ready to go.

4. More time should have been devoted to analyzing program listings on paper. Perhaps a criterion level of mastery could be established at such a task before "hands-on" time was granted.

5. After a period of instruction involving programming concepts common to all, students should be able to choose an area of interest or concentration for individualized study (units could be contained on disks, and supported by media such as audio tapes with headphones).

6. Classrooms containing computers should be organized so that there is one desk per student and the computers should be arranged so that they are behind the students when they are receiving instruction.

7. Even small classrooms should be equipped with large monitors located strategically around the room for demonstration purposes.

8. Programming instruction should involve discovery exercises as well as demonstration exercises.

9. Since the score in LA was a predictor of success in programming in this study, students in this area should be given more opportunities to work with microcomputers and related software, instead of having access only in math/science or business-related applications.

Recommendations for Future Research

The focus of this study was to determine the predictor variables for success in computer programming. To go one step further and determine how best to structure a program-

ming course for students who were <u>unsuccessful</u> in this study is, although essential, beyond the scope of this study. I would, therefore, prefer to consider this study as a pilot for a larger study investigating a comprehensive curriculum. Eventually, all students should arrive "at the same destination" in programming skills, however dissimilar their "travel routes". Could this study be replicated and improved by basing it on a larger sample of students in both rural and urban schools, in junior high/senior high and in elementary/junior high schools? The students involved in this study were "first-time" programmers; would the same generalizations apply to students with several years of "hands-on", although not necessarily programming, experience?

As the <u>Modern Language Aptitude Test</u> has proven reliable in predicting success in learning a second language, should <u>it</u> be investigated for inclusion in any battery for predicting success in learning a programming language (Pimsleur, et al., 1962; McEwan, J., 1974; Carroll and Sapon, 1967)?

Kline, in discussing aptitude tests (1976), points out that such tests may tell more about the particular job than people, as it is evident that an aptitude for programming is highly unlikely to be an inheritable ability. Nevertheless, an aptitude test designed specifically to be included in a programming course at the secondary level would be useful to identify immediately those students likely to experience

difficulties before they become frustrated and "turned off". Also, one instrument including aspects of aptitude identified in this study would be much less expensive and time-consuming to adminster.

Based on the results of such an instrument and the identification of the learners' cognitive styles, a logical next step would be to design the instruction to provide for individual differences. If the teaching of a programming language were based on an interactive microcomputer-based delivery system for instance, branched adaptations might involve variations in informational chunk sizes, response time and type, sequencing, and choice of graphic screens and instruction, as well as providing for audio, video or print support for self-instruction peer tutoring and remediation (Smith, P.L., 1984). Ausburn and Ausburn (1978) refer to planned supplantation involving alteration of the task requirement causing difficulty. In emphasizing that problems with instruction should be considered as instruction-based rather than learner-based (p. 342) for the purposes of designing alternate approaches, they distinguish between compensatory and conciliatory supplantation where compensatory supplantation provides specific processes that the learner cannot himself provide, and where conciliatory supplantation capitalizes on the use of instructional modes preferred by the learner. Either of these approaches implies a many-step process that begins with identifying the learner characteristics relevant to the task; a step that

may be accomplished by a "predictor instrument" such as the researcher suggests be developed.

Finally, it seems appropriate to conclude on a note of caution from "Computers and Equity":

> The most critical problem we'll face by pushing for computer literacy is a widening split of the already strained divisions between social and economic classes. ... The ghetto child won't have quite the same opportunities for exposure to computers - his gateway to computer literacy will be closed. Not only is this morally objectionable, it is a potential catalyst for a social upheaval we may not survive. ... It is imperative that we not blindly pursue this goal of computer literacy. We must closely examine the directions in which the Information Age is sweeping us and avoid conflicts with which we cannot cope. Computer literacy can mean significant and valuable alterations to our future, but it will be a positive development only when it is available to all.

REFERENCES

Ausburn, L.J. and Ausburn, F.B. Cognitive Styles: Some Information and Implications for Instructional Design. ECTJ, 26, No. 4, 337-354, Winter 1978.

Brown, H.D. Principles of Language Learning and Teaching. New Jersey: Prentice-Hall, Inc. 1980.

Budner, S. Intolerance of ambiguity as a personality variable. Journal of Personality, 39: 29-50, 1962.

Carroll, J.B. Research on teaching foreign languages. In: Handbook of Research on Teaching. Edited by N.L. Gage, 1963. pp. 1060-1100.

Carroll, J.B. Characteristics of successful second language learners. In: Viewpoint on English as a Second Language. Regents Publishing Co. Inc., 1977.

Carroll, J.B. and Sapon, S.M. Elementary Modern Language Aptitude Test. The Psychological Corporation, New York, 1967.

Cheney, P. Cognitive Style and Student Programming Ability: An Investigation. AEDS Journal, 288-291, Summer 1980.

Computer Literacy Steering Committee. Computer Literacy Report and Recommendations. Alberta Education, January 1982.

Dulay, H. and Burt, M. Remarks on creativity in language acquisition. In: Viewpoints on English as a Second Language. Edited by M. Burt, H. Dulay and M. Finnochiaro. Regents Publishing Co. Inc., New York, 1977.

Feldman, J.A. Programming Languages. Scientific American, 241:6, 94-116, December 1979.

Fisher, G. Developing a District-Wide Computer Use Plan. The Computing Teacher, January 1983.

Goddard, W.P. and Wright, A.E. Educational Computing, Discussion Paper 10/79. In: Computer Literacy in the Schools of B.C.. Ministry of Education, Science and Technology, Province of British Columbia, October 1979.

Harding, F.D. Tests as selectors of language students. Modern Language Journal, 42, 120-122, 1958.

Hopkins, K.D. and Glass, G.V. Basic Statistics for the Behavioral Sciences. Prentice-Hall, Inc., New Jersey, 1978.

Howell, M.A., Vincent, J.W. and Gary, R.A. Testing aptitude for computer programming. Psychology Reports, 20, 1251-1256, 1967.

Hull, T., Holt, R.C. and Phillips, C. Teaching Computer Science. Ontario Ministry of Education, 1975.

Hysmans, J.H. The effectiveness of the cognitive style constraints in implementing operation research proposals. Management Science, 17, September 1970.

Johnson, D.C., Anderson, R.E., Hansen, T.P. and Klassen, D.L. Computer Literacy and Awareness. In: Microcomputers in Secondary Education: Issues and Techniques. Kogan Page, London, 1981.

Kline, P. Psychological Testing: The Measurement of Intelligence, Ability and Personality. Malaby Press, London, 1976.

Krashen, S.D. The monitor model for adult second language performance. In: Viewpoint on English as a Second Language. Edited by M. Burt, H. Dulay and M. Finnochiaro. Regents Publishing Co., New York, 1977.

Krashen, S.D. Second Language Acquisition and Second Language Learning. Pergamon Press, Ltd., Oxford, 1981.

Lemos, R.S. Measuring programming language efficiency. AEDS Journal, 261-273, Summer 1980.

Lusk, J. and Kersnick. The effect of cognitive style and report format on task performance: The MIS consequences. Management Science 25:8, 787-798, August 1969.

McEwan, J.M. The Effectiveness of the Elementary Modern Language Aptitude Test and the Seashore Measures of Musical Talents in Predicting Grade 7 Oral and Written French Scores. Colloquium, University of Alberta Press, 1974.

Moursund, D. Teaching Basic and Fortran, January 1973.

Naiman, N., Frolich, M., Stern, H.H. and Todesco, N. The Good Language Learner. Research in Education, Series I, The Ontario Institute for Studies in Education, 1978.

Oltman, P.K., Rasking & Witkin, H.A. and Karp, S.A. Group Embedded Figures Test. Consulting Psychologists Press Inc., Palo Alto, California, 1971.

Peutz, W. Computers and equity. Microgram: The Computing Teacher, March 1983.

Pimsleur, P., Stockwell, R.P. and Comrey, A.L. Foreign language learning ability. Journal of Educational Psychology, 53, 15-26, 1962.

Ragsdale, R.G. Computers in the School. The Ontario Institute for Studies in Education, 1982.

Sammet, J.E. Programming Languages - History and Fundamentals. Prentice-Hall Inc., New Jersey, 1969.

Smith, P.L. Cognitive Styles Research - Implications for Instructional Design? Paper presented at the Annual Convention of the Association of Educational and Communications Technology, Dallas, 1984.

Thorndike, R.L. and Hagen, E. Canadian Cognitive Abilities Test. Multilevel Edition, 1974.

Watt, D.H. Computer literacy: What should schools be doing about it? In: Microcomputers in the Schools. Oryx Press, Phoenix, Arizona, 1981.

Zachmeir, W.J. K-8 computer literacy curriculum. The Computing Teacher, March 1983.

APPENDIX A

Budner's Scale of Tolerance of Ambiguity

INTOLERANCE OF AMBIGUITY

	Designed to tap[a]	
	Type of Response	Type of Situation
Positive items:		
1. An expert who doesn't come up with a definite answer probably doesn't know too much	PD	I
2. There is really no such thing as a problem that can't be solved	PD	I
3. A good job is one where what is to be done and how it is to be done are always clear	OS	C
4. In the long run it is possible to get more done by tackling small, simple problems rather than large and complicated ones	OS	C
5. What we are used to is always preferable to what is unfamiliar	PS	N
6. A person who leads an even, regular life in which few surprises or unexpected happenings arise, really has a lot to be grateful for.	PS	N
7. I like parties where I know most of the people more than ones where all or most of the people are complete strangers.	PS	N
8. The sooner we all acquire similar values and ideals the better.	OD	C
Negative items:		
9. I would like to live in a foreign country for a while.	PS	N
10. People who fit their lives to a schedule probably miss most of the joy of living.	PS	C
11. It is more fun to tackle a complicated problem than to solve a simple one.	PS	C
12. Often the most interesting and stimulating people are those who don't mind being different and original.	PS	C
13. People who insist upon a yes or no answer just don't know how complicated things really are.	PD	C
14. Many of our most important decisions are based upon insufficient information.	PD	I
15. Teachers or supervisors who hand out vague assignments give a chance for one to show initiative and originality.	OS	C
16. A good teacher is one who makes you wonder about your way of looking at things.	OS	C

[a] Codes are as follows:

Type of response	Type of situation
PD - phenomenological denial	I - insolubility
OS - operative submission	C - complexity
PS - phenomenological submission	N - novelty
OD - operative denial	

APPENDIX B

Cupertino School District

Computer Literacy and

Computer Use Framework

our experiences in teaching children to use computers, we include these objectives with the knowledge that most children, given ample time, can achieve them.

We realize that teacher expertise plays an important role in students' learning. Each school must assess the strengths of its staff, the configuration of computers, and their locations and the priority of usage. This curriculum is a flexible document with suggested grade levels for skills to be introduced (I), expanded (E) or reinforced (R). Schools can follow the specific grade levels suggested or the grade blocks (K-3, 4-6, 7-8) to guide their instructional plan.

COMPUTER AWARENESS

The student will:	K	1	2	3	4	5	6	7	8
1. Recognize the makeup of a computer.									
101. Identify parts of a computer.	I	E	R	R	R	R	R	R	R
102. Use the microcomputer in the school.	I	E	E	E	E	R	R	R	R
103. Use a prepared program in a microcomputer.	I	E	E	E	E	R	R	R	R
104. Understand computer terms.		I	E	E	E	E	R	R	R
105. Use appropriate terms when talking about computers		I	E	E	E	E	R	R	R
106. Define software and hardware.			I	E	E	R	R	R	R
107. Explain how a computer works (input/output).		I	E	E	E	R	R	R	R
108. List different computer languages and their uses.		I	R	R	R	R	R	R	R
109. Recognize capabilities of different computer types (sizes, brands and special-purpose machines).				I	E	E	E	E	R
2. Describe how computers affect our lives.									
201. Describe common uses of computers.				I	E	E	R	R	R
202. Explain ways computers affect people's lives.			I	E	E	E	R	R	R
203. Identify computer-related occupations.			I	E	E	E	R	R	R
204. List values of computer skills for future employment.						I	E	R	R
205. Explain the statement "Computer mistakes are mistakes by people."							I	E	R
206. Identify common tasks which are not suited for computers					I	E	E	R	R
207. Hypothesize about the limitations of computers.								I	E
3. Trace the history of computers.									
301. List the characteristics of each generation of computers					I	E	E	E	R
302. Learn the history of Silicon Valley.				I	E	E	E	R	R
303. Describe the impact of computers on Silicon Valley: social, political and environmental.							I	E	R
4. Understand the moral issues involved with computer use.									
401. Understand implications of copyright laws.		I	E	E	E	I	E	E	I
402. Identify advantages and dangers of computer data bases								I	E
403. Describe the serious legal issues resulting from wide-spread computer use	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

COMPUTER INTERACTION SKILLS

The student will:	K	1	2	3	4	5	6	7	8
5. Develop keyboard skills.									
501 Identify and use letters and numbers on a keyboard	I	R	R	R	R	R	R	R	R
502 Identify and use common special purpose keys		I	E	R	R	R	R	R	R
503 Develop correct keyboarding skills				I	E	E	E	R	R
6. Demonstrate general uses of the computer									
601 Demonstrate proper care of software-hardware	I	R	R	R	R	R	R	R	R
602 Demonstrate how to insert the disk, turn on the computer and boot a program		I	R	R	R	R	R	R	R
603 Demonstrate ability to stop, escape from and continue a program as needed				I	R	R	R	R	R
604 Run a program from the catalog or a menu			I	R	R	R	R	R	R
605 Recognize simple error messages (syntax error, break in, etc.)		I	E	E	R	R	R	R	R
7. Demonstrate specific uses of the computer.									
701 Use and interact with a drill and practice program.			I	E	E	R	R	R	R
702 Use and interact with a simulation program				I	E	E	E	R	R
703 Use and interact with a problem-solving program.						I	E	E	E
704 Use a computer as a word processor.						I	E	E	E
705 Use a data base program								I	E
706 Use an accounting program (e.g. VisiCalc TM)								I	E
707 Use a utility program								I	E
8. Know basic programming strategies.									
801 Describe standard flowchart symbols			I	E	E	R	R	R	R
802 Read a flowchart			I	E	E	E	R	R	R
803. Predict computer output given a program list					I	E	E	R	R

COMPUTER PROGRAMMING SKILLS

The student will:	K	1	2	3	4	5	6	7	8
9. Perform basic programming skills.									
901 Explain the concept of programming			I	E	E	E	E	R	R
902 Apply problem-solving strategies effectively using a computer program			I	E	E	E	E	E	E
903 Perform simple procedures using DOS commands to:									
903.1 Copy a simple program						I	R	R	R
903.2 Save a simple program				I	R	R	R	R	R
903.3 Delete all or part of a program						I	R	R	R
904 Explain simple error messages		I	E	E	E	R	R	R	R
905 Use editing procedures to correct programs				I	E	E	E	E	E
10. Perform specific programming skills.									
1001 Develop and apply strategies for debugging programs				I	E	E	E	E	E
1002 Write a flowchart to represent a solution to a task					I	E	E	R	R
1003 Translate a simple flowchart [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

The student will	K	1	2	3	4	5	6	7	8
Perform simple tasks using the language Logo									
1101 Write simple procedures in Logo	I	I	I	I	I	E	I	I	I
1102 Create graphic designs	I	E	I	I	I	E	E	E	E
1103 Write simple text programs (e.g. simple quiz, dialog)		I	I	I	I	I	I	E	I
1104 Write procedures with single and multiple inputs		I	I	I	E	I	E	E	I
1105 Write procedures using recursion		I	I	E	I	E	I	I	I
12 Perform simple tasks using the language PILOT.									
1201 Write simple programs in PILOT		I	E	E	E	I	E	R	R
1202 Create graphic designs		I	E	E	E	E	E	R	R
1203 Write simple text programs (e.g. quizzes, dialogs, conquains)					I	E	E	R	R
1204 Use relative and absolute coordinates in graphics programs		I	E	E	I	E	R	R	R
1205 Use absolute coordinates in graphics programs					I	E	E	E	E
1206 Use music in a program					I	E	E	R	R
13. Perform simple tasks using the language BASIC.									
1301 Recognize variations							I	R	R
1302 List fundamental statements and commands								I	E
1303. Distinguish random computer commands from a computer program								I	R
1304 Distinguish system commands from program statements								I	R
1305 Write a simple program in text and graphics								I	E
1306 Write a simple program to accomplish a specific task in BASIC								I	E

SOCIAL SCIENCES

The student will:	K	1	2	3	4	5	6	7	8
SS-1. Describe how computers affect our lives.									
SS-101 List several ways that computers are used in everyday life	I	E	E	E	E	E	E	E	E
SS-102 Identify ways that computers are used to help consumers					I	E	E	E	E
SS-103 Illustrate the importance of the computer in modern science and industry					I	E	E	E	E
SS-104 Identify career fields related to computer development and use					I	E	I	E	E
SS-105 State the value of computer skills for future employment								I	E
SS-106 Define the term "data base" and its use in various careers								I	E
SS-2. Recognize the legal and moral issues involved in computer use.									
SS-201 Describe problems related to the "invasion of privacy."								I	E
SS-202 Describe some advantages and disadvantages of a data base of personal information								I	E
SS-203 Describe ways in which computers are used to commit a wide variety of crimes and how these crimes are detected								I	E
SS-3 Describe how social scientists use computers.									
SS-301 Describe how computers are used by sociologists and other social scientists								I	E
SS-302 Describe how computer simulations are used in problem-solving situations					I	E	E	R	R
SS-303 Identify ways in which computers help make decisions.								I	E
SS-304 Explain how computer graphics are used in engineering, science, art, etc.								I	E
SS-305 Explain how computers are used as devices for gathering and processing data								I	E
SS-306 List several sampling techniques and statistical methods used in the social sciences.								I	E
SS-307 Describe computer applications such as those consisting of information storage and retrieval, process control and aids to decision making computation and data processing and simulation and modeling								I	E

LANGUAGE ARTS

The student will:	K	1	2	3	4	5	6	7	8
LA-1. Apply language arts skills to computer uses.									
LA-101 Define (and spell) basic computer terms.	I	E	E	E	E	E	E	R	R
LA-102 Explain the meaning of "word processing."							I	E	E
LA-103 Use a computer as a word processor.							I	E	E
LA-104 Describe some of the ways computers are used in the information and language-related careers.								I	E
LA-105 Edit a simple program for spelling, punctuation and/or syntax errors					I	E	E	E	E

SCIENCE

The student will:	K	1	2	3	4	5	6	7	8
S-1. Describe and define the computer and its processes.									
S-101 Describe a computer and how it works.	I	E	E	E	E	E	E	R	R
S-102 Describe the historical development of computing devices as related to other scientific devices.					I	E	E	R	R
S-103 Differentiate among micro, mini-, and mainframe computers and identify the five major components of any computer					I	E	E	R	R

The student will:	K	1	2	3	4	5	6	7	8
S-104 Define "input" and output" and give an example of each					I	E	E	R	R
S-105. Recognize the relationship of a program and input to the output.								I	E
S-106. Explain the basic operation of a computer system in terms of the input of data or information the processing of data or information and the output of data or information.								I	E
S-107 Recognize the need for data to be organized to be useful and relate this to its application to computers.								I	E
S-108 Describe how computers process data (searching, sorting, deleting, updating, summarizing, moving, etc.)								I	E
S-109 State what will happen if instructions are not properly stated in the precise language for that computer.					I	E	E	R	R
S-2. Explain how scientists use computers.									
S-201. Describe the computer's place in man's growing understanding of science.								I	E
S-202. Show how a scientist would use a computer.								I	E
S-203. Explain how computers are used in predicting, interpreting and evaluating data.								I	E
S-204. Explain how computers are used in testing and evaluating hypotheses								I	E

MATHEMATICS

The student will:	K	1	2	3	4	5	6	7	8
M-1. Describe how the computer is based on standard logic patterns.									
M-101. Explain that a computer design is based on standard logic patterns.								I	E
M-102. State the meaning of "algorithm."								I	E
M-103. Explain what is being accomplished by a given algorithm.								I	E
M-104. Follow and give correct output for a given algorithm.								I	E
M-2. Describe how a computer could be used to accomplish logical or arithmetic tasks.									
M-201. Describe the standard flowchart symbols					I	E	E	R	R
M-202. Draw a flowchart to represent a solution of a proposed problem					I	E	E	R	R
M-203. Order specific steps in the solution of a problem					I	E	E	R	R
M-204. Differentiate between analog and digital devices								I	E
M-205. Translate mathematical relations and functions into a computer program								I	E
M-206. Use the computer to accomplish a mathematical task.					I	E	E	R	R
M-207 Evaluate output as to its reasonableness in terms of the problem to be solved and the given input					I	E	E	R	R
M-208 State that data must be organized to be useful								I	E
M-209 Describe the techniques computers use to process data (searching, sorting, deleting, updating, summarizing, moving, etc.)								I	E
M-210 List several ways computers are used to process statistical data								I	E

END■

A COMPUTER USE FRAMEWORK

Use	Programming	CAI	Parts	Impact
		GRADE K-5		
Operate Load program Respect copyrights	Introduction to programming Use of pseudo-languages (PAK JANA, (BIG TRACK) Beginning languages: PILOT, Logo	Games Logic and problem solving	Identify major parts (input, operations) Vocabulary Basic functions	Pervasiveness Importance Career awareness, use and influence of computer
		GRADES 6-8		
Appropriate use, i.e. efficient, best tool for task Typing Keyboard and functions	Programming (BASIC) Modifying programs Problem solving	Remedial Simulations Word processing Logic and problem solving	Identify specific parts and functions: RAM, ROM, etc. Vocabulary Software formats (media)	Vocational uses and impacts Equals (sex-equity in availability and opportunity) Computer-related jobs Impact
		GRADES 9-12		
Appropriate programs Vocational use (word processing, data base, network, telecommunications) (Different hardware)	Advanced programming (BASIC, Pascal, machine, assembly, etc.) Ability to transfer and and modify programs	Remedial Simulations Tutorial Word processing Using data bases Logic and problem solving Vocational uses	(In electronics, logic and operations) Vocabulary Evaluation and selection (hardware and software)	Social questions (Job dislocation, isolation, computer crime, privacy, impact on nature of work) Computer-related jobs and job skills

APPENDIX

(Excerpts from Report to the Board)

TO: Board of Education

FROM: Steve Goldstone, Superintendent
Dick Rosenquist, Assistant Superintendent

RE: Presentation of Computer Plan and Computer Demonstration

BACKGROUND INFORMATION

The written report of the Computer Planning Committee is being submitted to the Board by inclusion in the Board packet for March 9th. The report contains a curriculum entitled *Computer Literacy* for pre-school through grade twelve and a 3-year budget calendar. Members of the parent-teacher-administrator Computer Planning Committee will be on hand to contribute to the oral discussion.

Most people, I believe, accept that microcomputers are immensely important to our society in general and computer literacy has become important to students preparing for college and for careers. The problem we all share is grasping the potential of computers for education, as today's microcomputers are amazingly versatile, powerful machines. Yet most of us know little about them.

To show Board members what a "second generation" microcomputer can do, I've arranged for a demonstration which involves a new computer (Monroe) which is *representative of a number of others recently put on the market*. The presentation is not designed as a sales pitch for that particular computer but rather to demonstrate the abilities of good microcomputers.

PROPOSAL

Acceptance of the Computer Planning Committee report for inclusion into the budget process for 1982-83.

RECOMMENDED ACTION

Information only.

APPENDIX C

Summary of Lessons

Day One

<u>Memory</u>

-instructions and data are stored here
_one bit is the tiniest possible unit of information
-there are 8 bits in 1 byte in the Apple
-1K=1 kilobyte
-1K=1024 bytes

<u>ROM</u>

-read only memory
-communicates instructions (BASIC) to the computer in language that the
  computer can understand
 -power off doesn't affect ROM

<u>RAM</u>

 -random access memory
 -or, read/write memory
 -only in use when the power is on

<u>DOS</u>
 -
-disk operating system
 -program that controls the disk functions
 -loading a copy of DOS into the computer is called "booting the system"
or "booting up"

<u>Initializing a Disk</u>

 -formatting=organizing the disk into sectors (usually 16)

NOTES

DAY TWO

<u>Initializing a Disk</u>

-follow these steps:

1) boot up the system with an initiallized disk
2) remove the disk used in #1
3) insert the blank disk
4) write a short program

```
eg. NEW
    10 HOME
    20 PRINT "COMPUTER LITERACY OPTION"
    30 END
```

5) RUN the program
6) INIT HELLO

-the disk is initiallized when the light on the disk drive goes out and the cursor and prompt appear in the HOME position

<u>CATALOG</u>

-this is a list of all the programs available to you on this disk
-I=Integer Basic
-A=Applesoft Basic
-T=Text File
-B=Binary Program
-*=a locked file
-006=disk sectors needed to store this program

<u>LOAD</u>

-type LOAD,NAME,RETURN

<u>RUN</u>

-RUN will start the program
-if you want to read a program only,just RUN it first

<u>Crtl-RESET</u>

-use this procedure to exit a program only if you do not want to SAVE

QUESTIONS

1) When initiallizing a disk;
   - why use the command NEW?
   - what should your first line be?
   - what should your last line be?
   - does INIT HELLO have a line number?
   - what is happening on the blank disk?

2) When you boot the system what are you actually doing?(hint;DOS)

3) How do you get access to:
   -*B 006 BRIAN'S THEME ?

NOTES

DAY THREE

Programming
-this means to organize a set of statements into a "unit"
-there ate two types of programming:

1)Immediate Mode
-similar to a calculator
-no program lines
-statements are executed immediately

```
eg. PRINT 3 + 4
    RETURN
    7
```

2) deferred Mode
-programming mode
-program line numbers
-statements are stored in memory

```
eg.10PRINT "I LOVE MICROCOMPUTERS"
    RETURN
    RUN
    I LOVE MICROCOMPUTERS
```

Some Useful Commands

NEW
-this command erases the contents of RAM
-begin every programming session with NEW
-does not clear screen or erase contents of disk

HOME
-make this the first ststement in your program
-clears the screen
-does not erase the contents of RAM

LIST
-use to see a list of all program statements
-entire list or portions of program

```
eg. LIST
    entire program
eg. LIST 20
    only line 20 is listed
eg. LIST 20-1000 (or LIST 20, 1000)
    lines 20 to 1000 are listed
eg. LIST 20- (or LIST 20,)
    lines 20 to END are listed
```

END
-always end your program with this statement

PRINT
-tells computer to display something
-computer will display everything inside the quotation marks

```
eg. PRINT "I LOVE MICROCOMPUTERS"
    RETURN
    I LOVE MICROCOMPUTERS
```

PRINT -continued

-the PRINT statement (or, ?) can perform a varied numbr of functions

-it is especially useful for formatting, or designing a screen display

- below is an example of a screen display done with PRINT statements

eg.

1) there are 40 columns and 24 lines on an APPLE text screen
2)plan your screen display by writing down exactly what will be displayed
3) use an APPLE text formatter sheet, if available
4) count the exact number if spaces in your line(s) of text: include spaces between words, all punctuation, apostrophes, etc.
5) subtract the number in step 4 from 40 (because there are 40 columns)
6) decide how many spaces you need on each side of the text by dividing the number you got in step 5 by 2
7) ?" sp. sp. sp. sp. ANDY MOOG IS VERY, VERY, AWESOME sp. sp. sp. sp. "
8) follow the same procedure to center the line in the middle of the screen.

QUESTIONS:

1) using your text formateer sheet, program on paper the following:

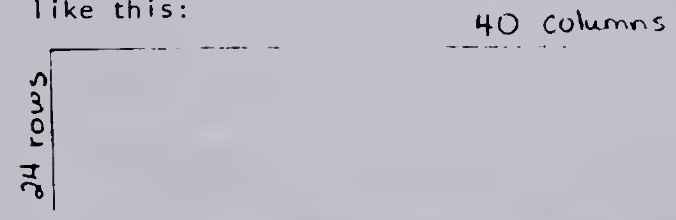
```
######################################
#                                    #
# MEET CANADA"S FUTURE PROGRAMMERS!  #
#                                    #
######################################
```

NOTES

DAY FOUR

THE SCREEN

- in text mode, which is what we've been working on so far, the screen looks like this:



-the cursor will be in the HOME position unless you specify where it is to start the display
-there are two ways to do this:
 1) by using the PRINT statement
    eg. ?:?:?:? (rows)
    eg. ?"   HI   "
 2) by using these two commands: VTAB, HTAB

VTAB

-this command tells the cursor to move down the screen a certain number of rows
-VTAB 10 positions the cursor on row 10, column 1
-avoid row 24 (VTAB 24), as this is where the cursor rests

HTAB

-this command tells the cursor to move over a certain number of columns
-HTAB 5 positions the cursor in column 5, on row 1

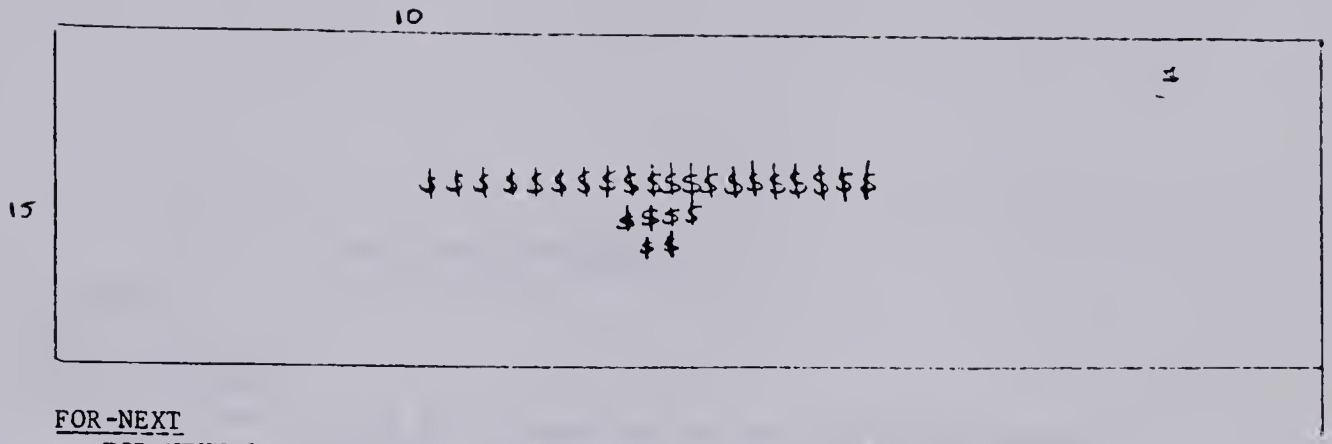
VTAB, HTAB

-use these commands together to position the cursor on a certain row, and in a certain column

```
     eg. VTAB 3:HTAB 3
         row 3, column 3
         VTAB 10, HTAB 5
         row 10, column 5
         VTAB 7, HTAB 11
         row 7, column 11
```

-obviously, these two commands are a lot more efficient than the PRINT command for this job!

QUESTIONS
1) what position is the cursor in?
   a) VTAB 6:HTAB 6
   b) VTAB 10
   c) HTAB 10



FOR-NEXT
-a FOR-NEXT loop will asso save you time and trouble
-use this command when you want a certain line of display repeated several times.
-a FOR-NEXT loop saves programming time, disk space, and memory

```
eg. 5 HOME
    10 VTAB 10:HTAB 10
    20 For I =1 TO 10
    30 ?"KENILWORTH SCHOOL"
    40 NEXT I
    50 END
```

-this program will print:

```
KENILWORTH SCHOOL
KENILWORTH SCHOOL
KWNILWORTH SCHOOL
KENILWORTH SCHOOL
KENILWORTH SCHOOL
KENILWORTH SCHOOL
KENILWORTH SCHOOL
KENILWORTH SCHOOL
KENILWORTH SCHOOL
KENILWORTH SCHOOL
```

-imagine how long it would have taken to program this line by line?!
-line 20 sets the counter (repeat line 10 times)
-line 40 sends the computer back up to line 20 until it goes through the loop 10 times

QUESTIONS

1) What will the following program RUN?

```
 5 HOME
10  VTAB 6
20  FOR I= 1 to 5
30 ?"########## IT WORKS! ##########"
40 NEXT I
50 END
```

2) Write a program that will display :

```
*****     THIS IS SO EASY     *****
```

10 times, starting at row 15

NOTES

LOW RESOLUTION GRAPHICS

1) to PLOT

PLOT column, row
eg. PLOT 23,18
plots a point at the 24th column and the 19th row on the screen. remember that the columns and rows in graphics mode are numbered from 0 to 39

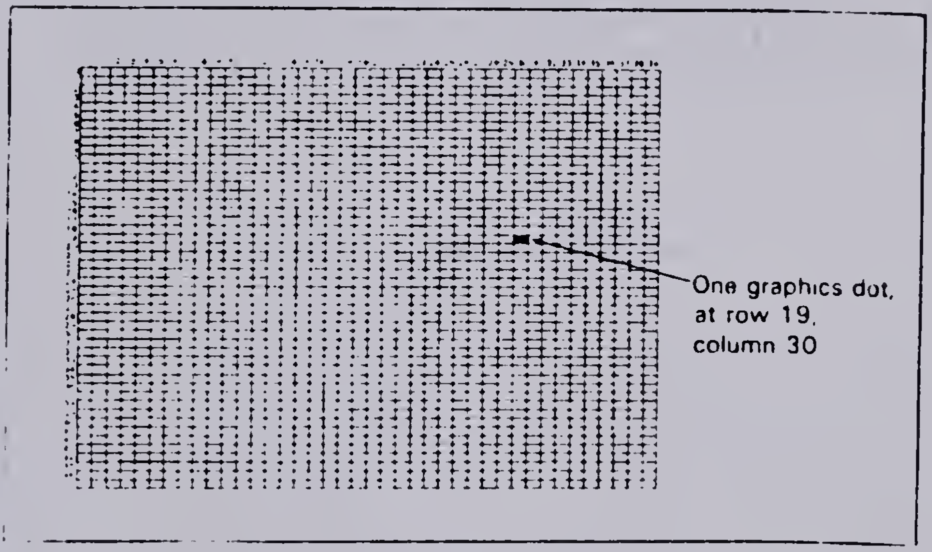
2) drawing lines

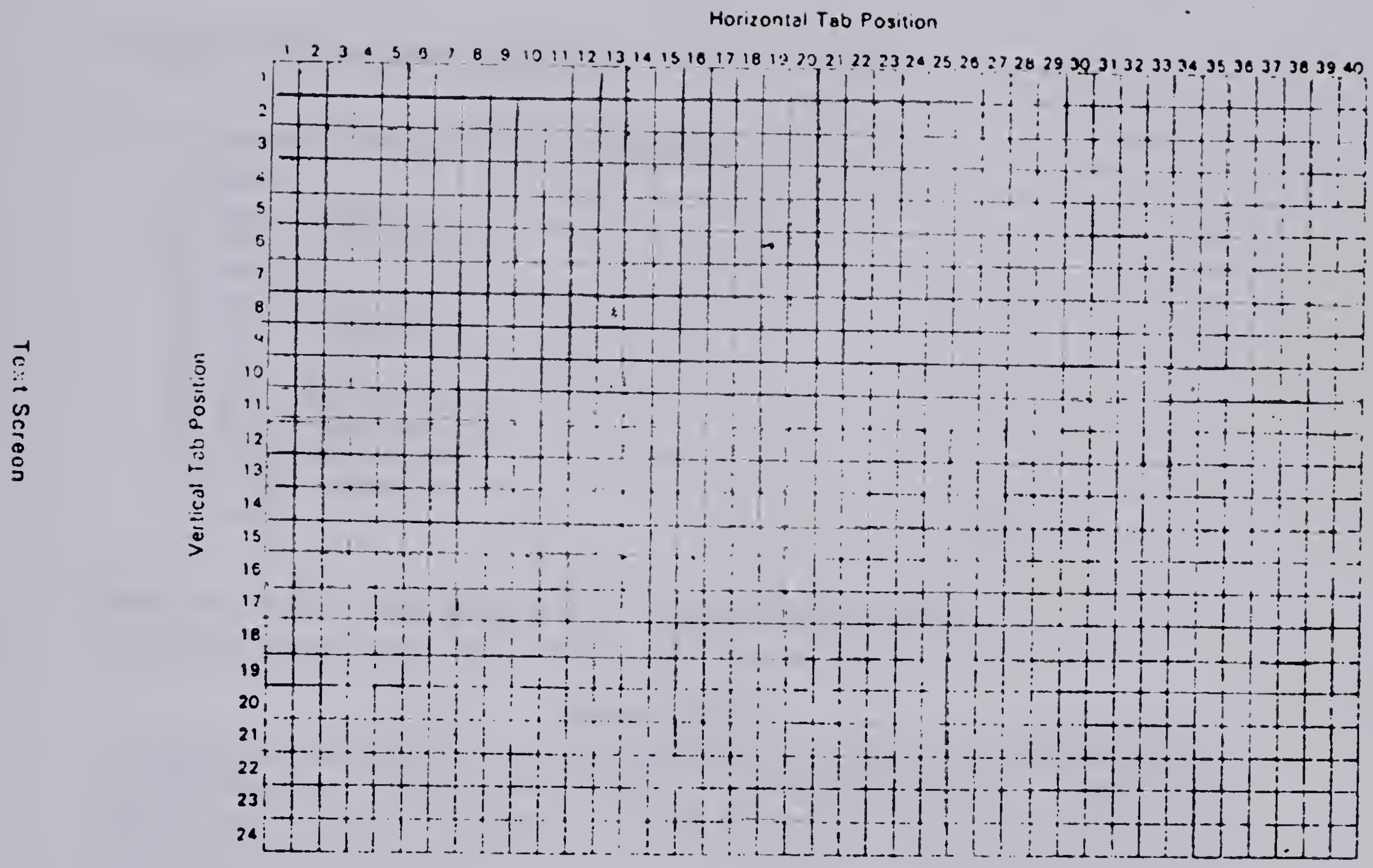
a) HLIN
eg. HLIN 0,39 at 12
draws a horizontal line from the left to the right (0-39) starting on row 12

b)VLIN

eg. VLIN 0,39 at 12

draws a vertical line from the top to the bottom of the screen (0-39) in column 12





PROGRAM PLANNING

```
10
 20
30
 40
 50
 60
 70
 80
 90
 100
 110
 120
 130
 140
 150
 160
```

REMEMBER: ALWAYS START WITH <u>NEW</u>, THEN <u>HOME</u>

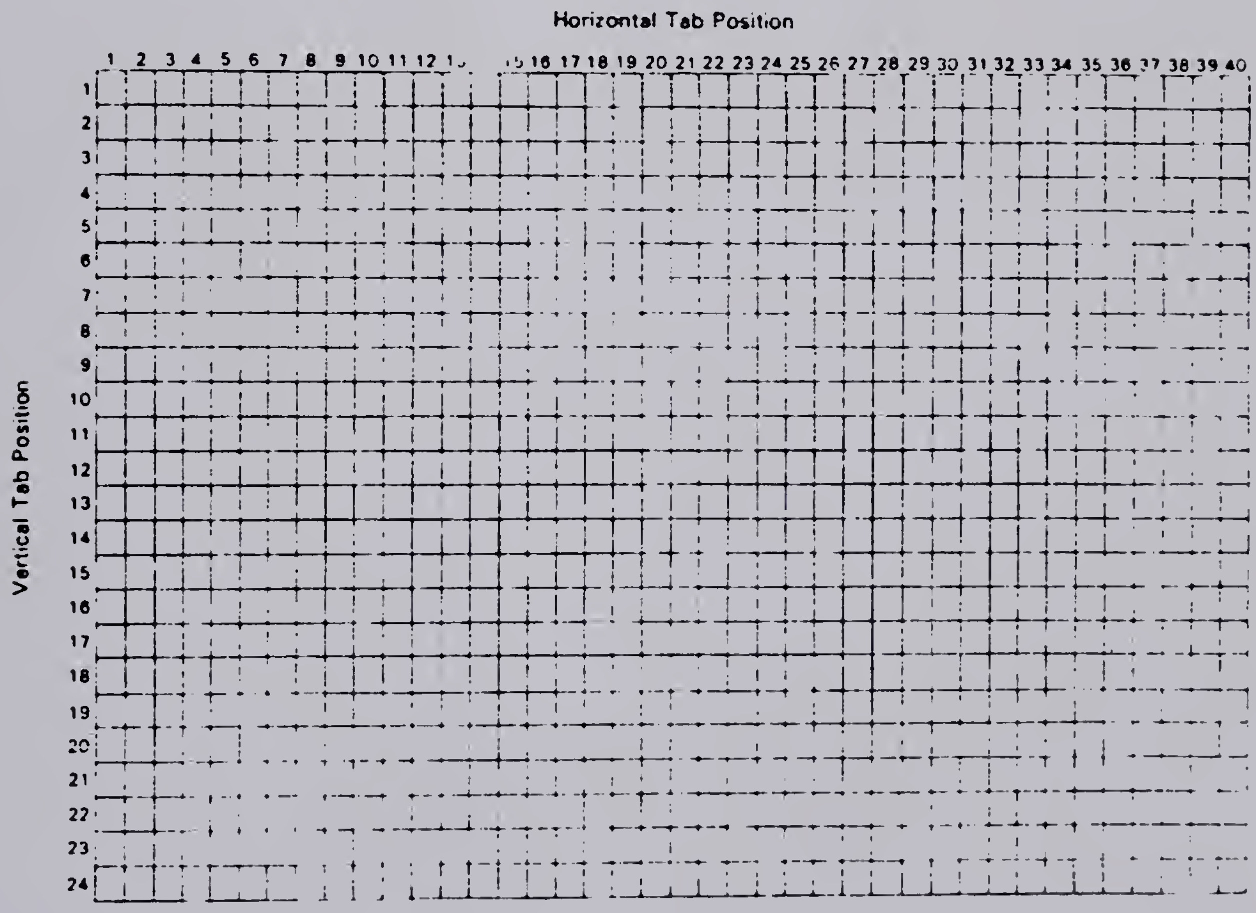
QUIZ

DO <u>ONE</u> OF THE FOLLOWING PROBLEMS:

1) A student wrote the follwing problem;

```
10 HOME
20 VTAB 5:HTAB 15
30 PRINT "**********"
40 VTAB 6
50 FOR I=1 TO 10
60 HTAB 15;PRINT "*         *"
70 NEXT I
80 VTAB 16:HTAB 16
90 PRINT "**********"
100 VTAB 10:HTAB 17
110 PRINT "THANKS!"
120 END
```

When the student types RUN, what will appear on the screen?
On the following grid, show exactly what will happen.



2) Write a program that will generate a screen display that will include;

a) a border of stars
b) two lines in the middle of the border, any message